

THESE

présentée à

L'UNIVERSITE PIERRE ET MARIE CURIE (PARIS VI)

pour l'obtention du

DOCTORAT TROISIEME CYCLE

Spécialité: MATHÉMATIQUES

Mention: INFORMATIQUE

par Madame Anne VILNAT

Sujet de la thèse: L'élaboration d'interventions pertinentes
dans une conversation homme-machine

Soutenue le 14 Décembre 1984 devant la Commission

composée de: Président M. J.C. SIMON

Examineurs: Mme A. BORILLO

MM. J.P. HATON

D. KAYSER

G. SABAH



THESE

présentée à

L' UNIVERSITE PIERRE ET MARIE CURIE (PARIS VI)

pour l'obtention du

DOCTORAT TROISIEME CYCLE

Spécialité : MATHEMATIQUES

Mention : INFORMATIQUE

par Madame Anne VILNAT

Sujet de la thèse : L'élaboration d'interventions pertinentes
dans une conversation homme-machine

Soutenue le 14 Décembre 1984 devant la Commission

composée de: Président M. J.C. SIMON

Examineurs: Mme A. BORILLO

MM. J.P. HATON

D. KAYSER

G. SABAH

Je remercie Monsieur le Professeur Jean-Claude Simon de m'avoir accueillie dans son équipe. Je lui suis reconnaissante d'avoir accepté de présider ce jury.

Je tiens à remercier Monsieur Gérard Sabah de m'avoir orientée vers ce domaine de recherche. Ses conseils et remarques m'ont toujours guidée au cours de ce travail.

J'exprime ma gratitude à Madame Andrée Borillo et à Messieurs Jean-Paul Haton et Daniel Kayser d'avoir accepté d'être examinateurs dans ce jury. Je leur suis reconnaissante de l'intérêt qu'ils ont ainsi montré pour mes recherches.

Tous les membres de l'équipe de Gérard Sabah m'ont aidée par leurs discussions et leurs remarques pertinentes, je tiens à les en remercier.

Je souhaite enfin exprimer mes remerciements à tous ceux qui ont contribué à l'achèvement de cette thèse, en particulier l'équipe de l'imprimerie de l'Institut de Programmation.

- INTRODUCTION

- CHAPITRE I : Comment "converser avec une machine" ? p. 15

I.1 - Historique des travaux sur le dialogue. p. 17

I.1.1 - Le dialogue comme activité. p. 18

a) Le dialogue dans les systèmes fondés sur la notion de stéréotype. p. 18

b) Le dialogue dans les systèmes-experts. p. 20

I.1.2 - Le dialogue comme objet. p. 22

I.2 - Le dialogue, phénomène social. p. 24

I.3 - Le dialogue dans un système de questions-réponses. p. 26

I.3.1 - Les thèmes développés au cours du dialogue. p. 28

I.3.2 - Le déroulement du dialogue. p. 30

I.3.2.1 - La relance du dialogue. p. 35

I.3.2.2 - L'explication. p. 37

I.3.3 - Exemples. p. 38

- CHAPITRE II : Les Représentations. p. 43

II.1 - Représentation interne des phrases. p. 45

II.2 - La représentation des connaissances. p. 47

II.2.1 - Représentation des connaissances syntaxico-sémantiques. p. 48

II.2.1.1 - Les connaissances morpho-syntaxiques. p. 49

II.2.1.2 - Représentation des concepts. p. 49

II.2.1.3 - Les relations entre mots et concepts. p. 52

II.2.2 - Représentation des connaissances pragmatiques. p. 54

II.2.2.1 - Le nom. p. 54

II.2.2.2 - La description. p. 54

II.3 - Représentation du raisonnement.	p. 55
II.3.1 - Présentation générale.	p. 55
II.3.2 - IMALGO : la représentation du module RAISON.	p. 58
II.3.2.1 - La structure de IMALGO.	p. 59
II.3.2.2 - Les connaissances représentées au niveau de IMALGO.	p. 60
II.3.3 - IMEXEC : La représentation d'une exécution du module RAISON.	p. 64
II.4 - La représentation du thème.	p. 67
II.5 - La représentation de l'utilisateur.	p. 71
II.5.1 - Les connaissances présupposées sur le domaine d'application.	p. 71
II.5.2 - Comment représenter le problème particulier d'un utilisateur.	p. 74
II.6 - L'historique du dialogue.	p. 76
II.7 - Conclusion	p. 78
- CHAPITRE III : Les procédures.	p. 79
III.1 - CONSTIM : création de IMALGO.	p. 81
III.1.1 - Le principe adopté.	p. 81
III.1.2 - La réalisation pratique.	p. 82
III.2 - Le suivi des représentations.	p. 86
III.2.1 - La création des arbres représentant le thème.	p. 86
1) Le thème introduit par l'utilisateur.	p. 87
A) L'intervention est une information.	p. 87
B) L'intervention est une demande.	p. 88
a) Demande objet.	p. 89
b) Demande causale.	p. 90
2) Les thèmes introduits par le système.	p. 92
III.2.2 - La prise en compte des connaissances de l'utilisateur.	p. 94
III.2.2.1 - Les connaissances sur le domaine d'application.	p. 94
III.2.2.2 - Définition du problème de l'utilisateur	p. 96
III.2.3 - La constitution de l'historique.	p. 97

III.3 - La procédure d'explication.	p. 100
III.3.1 - L'explication partielle.	p. 101
III.3.2 - L'explication générale.	p. 102
III.4 - La relance du dialogue.	p. 105
III.4.1 - Comment utiliser la représentation du problème.	p. 105
III.4.2 - L'utilisation des autres représentations.	p. 108
- CHAPITRE IV : Quelques exemples de dialogues.	p. 113
IV.1 - Exemple 1.	p. 115
IV.2 - Exemple 2.	p. 125
IV.3 - Exemple 3.	p. 131
- CONCLUSION	p. 137
- ANNEXES	p. 141
- BIBLIOGRAPHIE	p. 153

- INTRODUCTION -

Le terme de dialogue est employé dans les systèmes d'Intelligence Artificielle pour qualifier des comportements très variés.

Dès 1964, Raphaël considère que son système (SIR) est un programme qui "comprend" car, dit-il, "il fait preuve d'un comportement conversationnel comparable à celui d'un humain et semble posséder des capacités cognitives" (Raphaël 64, p.577).

En fait, les premiers systèmes réalisés dans le domaine du traitement automatique du langage avaient pour but essentiel de montrer qu'ils étaient capable de comprendre.

Une question se pose : comment faire la preuve de cette capacité ?

D'après Raphaël, toujours : "une hypothèse fondamentale est que la compréhension peut être démontrée par le dialogue".

C'est sur cette affirmation que se sont développés les travaux sur le dialogue. Nous avons cité Raphaël, mais on pourrait ajouter (Weizenbaum 66) ou (Winograd 72).

De là est né un malentendu sur l'utilisation de ce terme. Puisque la machine était capable de formuler une réponse relativement sensée à une question posée dans un langage plus ou moins naturel, on a pensé qu'elle "dialoguait" avec l'utilisateur. Le but de ces recherches n'était pas d'étudier le dialogue lui-même, mais de s'en servir pour prouver la validité du travail de compréhension. C'est ainsi qu'ont été évacués tous les problèmes propres au dialogue, tels qu'ils avaient été définis par les linguistes ou les psycho-linguistes.

Il s'agissait bien sûr d'une première approche, qui était intéressante compte tenu de l'état des recherches en Intelligence Artificielle, mais l'utilisation abusive de ce terme a longtemps masqué les vrais problèmes que pose le dialogue.

Il en est encore ainsi à la fin des années 70, où l'apparition de nouvelles méthodes de représentation des connaissances permet la création de systèmes plus intéressants, comme GUS (Bobrow 77), MYCIN (Shorliffe 76), ou encore Nudge (Goldstein 79) et PAL (Sidner 83). Dans tous ces systèmes, on trouve une composante de dialogue en langage naturel qui sert d'interface avec l'utilisateur. Mais la façon dont est mené le dialogue est fortement dirigée par le mode de représentation des connaissances utilisé : les "frames" en KRL pour GUS, les règles de production pour MYCIN, les "frames" en FRL pour Nudge.

Le dialogue n'est pas étudié pour les problèmes particuliers qu'il pose, il ne sert qu'à mettre en valeur une méthode de représentation des connaissances.

Prenant conscience des faiblesses de ces dialogues, d'autres études ont été entreprises, prenant le dialogue lui-même comme sujet de recherche. Elles ont tenté de montrer les qualités qu'un dialogue doit respecter pour être accepté par l'utilisateur : essayer d'adapter le système aux demandes de ce dernier au lieu de l'obliger à se plier aux caractéristiques du système. On peut citer dans ce cadre les

recherches de Hayes et Reddy (Hayes 79)), celles de Lehnert (Lehnert 79) et celles de Allen, Perrault et Cohen (Allen 80), (Perrault 78).

Reprenant les idées précédentes, et nous aidant des travaux de Grice sur le dialogue entre personnes (les principes régissant les conversations humaines (Grice 75)), nous avons essayé de définir les propriétés qu'un système doit respecter pour participer à un dialogue de façon acceptable.

Nous n'avons pas entrepris d'étudier ce phénomène dans son ensemble. Nous nous sommes intéressé au dialogue dans un système de questions-réponses destiné à être utilisé par le grand public, sans restriction sur le domaine d'application.

Dans ce cadre nous avons mis en évidence trois points principaux :

- la gestion du dialogue sur plusieurs phrases;
- la possibilité de poursuivre l'échange en cas d'échec de la recherche;
- la capacité d'expliquer les propres actions du système de raisonnement.

Reprenons plus en détails ces différents éléments.

1) Etre en mesure de mener un réel dialogue, sans se limiter à un simple échange composé d'une question de l'utilisateur à laquelle le système tente de répondre.

Nous considérons que dans un système de questions-réponses, l'utilisateur peut avoir besoin de plusieurs interrogations pour résoudre un problème, si ce dernier se révèle un peu compliqué. Elles peuvent concerner des sujets apparemment différents, mais liés dans le contexte particulier du problème.

Pour offrir cette possibilité, nous devons reconnaître les liens entre les thèmes évoqués et donc gérer d'éventuels changements de sujets au cours du dialogue, incluant des retours en arrière.

2) Pouvoir relancer le dialogue quand on ne trouve pas de réponse immédiate à une demande. Il est difficile d'admettre qu'un système, quand il a essayé de résoudre un problème, soit incapable d'aider l'utilisateur à préciser sa demande. En effet, le but d'un système, dit intelligent, est d'élargir sa recherche en essayant de produire diverses inférences, d'utiliser des notions de "synonymies",... Quand le système ne parvient pas à répondre, cela signifie qu'il a déjà essayé de trouver des résultats en appliquant diverses méthodes, mais qu'il est bloqué dans la poursuite de ses recherches. Il faut se servir du travail déjà réalisé par le système, pour proposer à l'utilisateur des options claires lui permettant de préciser efficacement sa demande. On peut également utiliser les connaissances du système sur le domaine d'application pour guider l'utilisateur dans ses choix.

3) Pouvoir expliquer les réponses que donne le système, quel que soit le mode de représentation des connaissances, et la méthode définie pour les utiliser.

Sans vouloir entrer dans la querelle procédural contre déclaratif, nous pensons qu'il faut pouvoir réaliser un dialogue cohérent dans l'un et l'autre cas. Le fait de pouvoir s'expliquer semblait jusqu'ici réservé aux systèmes déclaratifs. Mais il nous semble abusif de dire que les résultats obtenus par un système-expert peuvent être justifiés uniquement en fournissant la liste des règles utilisées, le critère de déclenchement de ces règles ayant lui aussi son importance.

Pour permettre la constitution de ces explications, nous avons dû munir notre système d'un "droit de regard" sur le travail qu'il effectue, d'une sorte de "conscience" de ce qu'il fait.

Nous avons appliqué notre travail à un système permettant la consultation des "pages jaunes" de l'annuaire téléphonique (Sabah 83). Nous avons introduit dans ce système une nouvelle composante destinée à gérer le dialogue avec l'utilisateur.

Au cours de ce travail, nous avons essayé de mettre en place des procédures indépendantes de l'application. Certaines options restent, bien sûr, liées au domaine particulier de cette étude; mais ce ne sont pas les caractéristiques fondamentales du système et il serait aisé de les modifier pour adapter notre travail à une autre application.

Tel que le système des pages jaunes était défini, il ne pouvait répondre qu'aux questions de l'utilisateur dont la forme sémantique peut se résumer ainsi :

- le numéro d'un M, M étant un métier;
- le numéro de quelqu'un qui V O, V étant une action sur un objet O.

Toutes les demandes de l'utilisateur sont interprétées comme des requêtes dont la réponse doit être cherchée dans les connaissances sur les professionnels figurant dans les pages jaunes.

A la suite de sa recherche, le système donne des réponses qui peuvent être :

- la rubrique sélectionnée, qui répond à la demande formulée;
- une liste de rubriques parmi lesquelles l'utilisateur doit choisir celle qui lui convient;
- une ou plusieurs rubriques, dont on est plus ou moins sûr qu'elles répondent à la demande.

L'utilisateur n'a plus aucune prise sur le dialogue. Il doit se contenter des réponses données ou formuler une nouvelle demande, pour laquelle on ne tiendra plus compte du travail précédent.

Nous avons voulu sortir de ce cadre trop rigide, afin de rendre à l'utilisateur la possibilité d'orienter différemment le dialogue, de demander des explications, en particulier quand les réponses qui lui sont données ne sont pas sûres.

Dans le chapitre I, nous revenons plus en détail sur les différents travaux qui ont été réalisés sur le dialogue, aussi bien en informatique qu'en psycho-linguistique.

Puis nous présentons le schéma général du système que nous avons réalisé, et nous illustrons sur des exemples les différentes possibilités qu'il offre.

Dans le chapitre suivant, nous définissons les représentations utilisées. Nous commençons par la représentation interne des phrases, suivie des représentations syntaxiques, sémantiques et pragmatiques.

Enfin nous détaillons les représentations qui sont utilisées plus spécifiquement par les procédures de gestion du dialogue. Ce sont les représentations des deux interlocuteurs (le système et l'utilisateur) et celles qui permettent le suivi de l'évolution du dialogue.

Le chapitre III présente les procédures mises en place afin de gérer le dialogue, en respectant les principes mis en évidence dans le premier chapitre. Nous indiquons d'abord les procédures permettant de créer et de mettre à jour les différentes représentations, puis nous détaillons le fonctionnement des procédures d'explication et de relance du dialogue.

Dans le dernier chapitre, nous présentons trois exemples complets afin d'illustrer ces différentes procédures.

- CHAPITRE I -

Comment "converser avec une machine" ?

I.1 Historique de travaux sur le dialogue.

Les premiers systèmes d'Intelligence Artificielle réalisés ont rapidement repris un vieux mythe : faire parler une machine, dialoguer avec elle. C'est ainsi qu'un système tel qu'ELIZA (Weizenbaum 66) a connu un certain succès. Ils ont eu pour mérite de mettre en évidence leurs limites : pas de véritable dialogue, si ce n'est un écho de ce qu'ils viennent d'entendre, en utilisant un certain nombre de règles, surtout syntaxiques, qui leur permettent de varier les formules énoncées.

Le premier système qui pose vraiment les problèmes intéressants a été développé par Winograd avec SHRDLU (Winograd 72). Celui-ci parvenait effectivement à mener un dialogue avec son interlocuteur, en expliquant ses propres actes, ou en posant des questions lorsque les indications qui lui étaient fournies se révélaient insuffisantes.

Le principal défaut de ce système est de fonctionner dans un univers très restreint : un monde d'une douzaine de blocs. De ce fait, le dialogue se résume à des explications concernant des actions effectuées sur des blocs suffisamment distincts dans leur désignation pour qu'il y ait peu de problèmes d'ambiguïtés. Les méthodes appliquées pour réaliser un dialogue avec le robot SHRDLU, sont impossibles à transposer pour un univers réel, comme le montre (Dreyfus 79).

Par contre, les idées qui sont à la base du travail de Winograd concernant le dialogue sont toujours intéressantes.

En effet, bien que ce système soit assez ancien, il est le seul qui permette de mener un dialogue complet, qui ne se limite pas à un simple échange : une question - une réponse éventuellement précisée. Le robot se souvient de ses actions, et des raisons pour lesquelles il les a accomplies.

Les travaux traitant des dialogues qui ont été menés depuis peuvent être répartis en deux catégories, correspondant à deux points de vue différents :

- ceux qui n'ont pour but que d'illustrer les avantages d'un type de représentation des connaissances sur le monde, considérant le dialogue comme une activité;

- ceux qui cherchent à comprendre des dialogues écrits sans que le système soit l'un des participants, considérant le dialogue comme un objet.

1.1.1) Le dialogue comme activité.

Dans cette première catégorie, on peut ranger les systèmes développés dans les années 70. C'est alors qu'a été ressentie la nécessité d'utiliser des représentations adaptées pour fournir à la machine une connaissance du monde de tous les jours.

Pour simplifier, nous dirons que deux courants se dégagent principalement : les systèmes inspirés par la notion de "frame" (ou "script", ou "scénario") introduite par Minsky (Minsky 74), que nous traduirons par stéréotypes, et les systèmes experts venus à la suite du succès de Shortliffe avec MYCIN (Shortliffe 76).

a) Le dialogue dans les systèmes fondés sur la notion de stéréotype.

Ces dialogues se déroulent toujours dans un cadre bien défini : procurer des renseignements sur des réservations de places d'avion (GUS (Bobrow 77)), organiser des réunions (PAL et Nudge (Bullwinkle 77)(Goldstein 79)), ... Les actions à effectuer sont alors bien définies et en nombre restreint. Du fait de ces restrictions, la maîtrise du dialogue reste au système. Nous allons voir plus particulièrement le cas de GUS.

Le dialogue n'est pas explicitement dirigé par le système : l'utilisateur peut dire ce qu'il veut. Mais dans les interventions, ne sont pris en compte que les renseignements que le système recherche, à savoir l'horaire et le parcours que l'utilisateur souhaite emprunter. Le dialogue suit toujours une certaine trame établie à l'avance. Ce fait est dû au type de représentation choisi. Le but est de compléter un certain stéréotype, en l'occurrence une "réservation".

Les procédures de dialogue sont entièrement dirigées par une motivation : remplir les éléments manquants à la définition complète d'une réservation.

Elles ne sont pas généralisables pour un dialogue dans un univers différent. On pourrait éventuellement penser à combiner des procédures destinées à effectuer une réservation d'avion avec d'autres possibilités du même type : réservation de train, de chambre d'hôtel, etc..., mais ces éléments resteraient des modules disjoints : on est loin d'un univers réel où le cloisonnement en petits modules est impossible dès que les problèmes deviennent plus complexes.

Ce serait faire jouer à un système une caricature de rôle comparable à celui de Maître Jacques dans *L'Avare* de Molière : il doit enlever sa casquette de cocher pour parler cuisine.

On retrouve une situation analogue dans les autres systèmes de ce type qui ont été évoqués.

Nous reviendrons plus loin sur les travaux réalisés par Allen, Cohen et Perrault (Allen 80) (Perrault 78) qui s'intéressent plus aux problèmes liés au dialogue lui-même, et aux plans des utilisateurs.

W. Lehnert (Lehnert 77) s'est attachée également à montrer l'importance de la forme et non pas seulement du fond, des réponses fournies par un système de questions-réponses. Mais ses travaux les plus récents (Schank 79) (Lehnert 83) concernent plutôt l'étude de dialogues dans lesquels elle n'intervient pas et nous en reparlerons plus loin.

b) Le dialogue dans les systèmes-experts

Les systèmes-experts étaient primitivement destinés à des spécialistes dans des domaines déterminés. Le problème de l'interface semblait donc peu important dans les premiers systèmes développés. Mais très vite, l'utilisateur a éprouvé le besoin de demander une justification sur une réponse fournie par le système.

Une double motivation a poussé les concepteurs de ces systèmes à accéder à ce désir : convaincre l'utilisateur de la cohérence des solutions obtenues en exhibant les règles utilisées et vérifier la validité de la base de connaissances du système pour éventuellement en combler les lacunes.

Une amorce de dialogue a alors été mise en place (Shortliffe 76), permettant à l'utilisateur d'interroger le système sur la façon d'aboutir à une solution. Toutefois, cette explication reste limitée à la liste brutale des règles appliquées. Plus le système devient complexe, moins l'explication est claire, la liste des règles devenant trop longue et de ce fait illisible. C'est pourtant dans ce cas qu'une explication serait la plus utile.

Le problème d'interface s'est donc révélé crucial pour l'acceptation de ce type de systèmes par des utilisateurs, en général des spécialistes, qui n'en sont plus les concepteurs et sont relativement réticents à les utiliser. C'est ce qui a poussé Langlotz et Shortliffe à développer le système ONCOCIN (Langlotz 83). Nous allons l'étudier de plus près, car il est le plus récent, et à notre avis le plus complet dans ce domaine.

La constatation essentielle qu'ils ont faite est que, pour être accepté, le système doit être souple et "explicatif" (c'est-à-dire en mesure de s'expliquer). Afin de respecter le critère de souplesse, les auteurs ont choisi de saisir les entrées du système par l'intermédiaire d'une grille - questionnaire calquée sur le support papier auquel les

médecins sont habitués, en y apportant quelques améliorations. Cette solution semble justifiée dans ce domaine particulier bien délimité. Le langage naturel n'est pas toujours le meilleur moyen de communiquer des informations précises!

Pour le second point, le fait que le système soit explicatif nous paraît parfaitement justifié, par contre le moyen utilisé nous semble très restrictif. En effet, les questions qu'il est possible de poser sont rassemblées dans une liste dans laquelle le médecin doit choisir celle qui lui convient.

La raison invoquée par les auteurs est d'éviter à l'utilisateur la peine de taper sa question au clavier. Sur ce point, il nous semble tout aussi pénible de retrouver à l'intérieur d'une liste la question que l'on se pose : elle peut être formulée différemment, on ne la trouve pas toujours. Cette solution ôte tout caractère spontané au dialogue. On en revient à un système de menu que l'on parcourt trop vite parce qu'il est fastidieux. On y sélectionne alors une question qui n'est pas toujours celle que l'on voulait poser. Le système propose de détailler certains points, mais ne laisse pas la possibilité de l'interroger.

De plus le procédé n'est pas généralisable : la liste des questions qu'il est possible de poser n'est pas toujours fermée. La véritable raison de ce choix semble plutôt être l'économie d'un analyseur suffisamment souple et rapide.

Enfin, ce système ne permet pas de gérer un dialogue comportant plusieurs interrogations et autorisant des questions sur l'une quelconque de ses réponses.

Par contre, le principe adopté pour répondre aux questions nous paraît très intéressant, dans la mesure où il essaie de ne pas être trop bavard : il ne fournit que les points essentiels. Nous verrons plus loin en quoi, dans un cadre différent, nous avons utilisé le même principe.

BLAH (Weiner 80) est un autre système dont le fonctionnement est fondé sur l'utilisation d'un corpus de règles. Weiner s'est intéressé également à la mise en place de procédures d'explication, en utilisant une structure arborescente. Mais dans ce problème il a étudié plus précisément la génération d'une structure externe compréhensible ce qui n'était pas simple vu le domaine d'application : la législation sur les impôts aux Etats Unis.

1.1.2) Le dialogue comme objet.

Parallèlement à ces systèmes de dialogue, des systèmes informatiques ont été développés afin de comprendre des dialogues qui se trouvent dans un texte. Nous nous sommes intéressés à ces systèmes pour les points d'étude théorique qu'ils présentent, et nous allons les évoquer brièvement.

Certains travaux ont pour but de résoudre les problèmes d'anaphores, d'ellipses, de référents, de choix d'articles (Grosz 77), (Sidner 83), (Webber 78), et leur façon d'aborder le problème du dialogue est complètement différente de la nôtre : il ne s'agit plus d'un échange d'information entre deux personnes, mais d'un point très particulier de cet échange.

Reichmann (Reichmann 78) s'est intéressée à la façon dont les participants à une conversation s'attachent à en respecter la cohérence. Elle insiste sur l'environnement dans lequel une conversation se déroule, et sur les procédures qu'utilise chaque participant pour identifier le modèle du discours d'un autre.

Dans l'équipe de Schank, W. Lehnert (Lehnert 83) (Schank 79) s'est particulièrement intéressée à des dialogues dans un cadre défini : des discussions sur le divorce.

Dans ces travaux, elle met en évidence tous les niveaux d'interprétation que l'on peut trouver dans un dialogue : du niveau simplement syntaxique à une interprétation "psychique" des personnages. Elle détaille, dans ces dialogues, jusqu'à quatorze niveaux d'interprétation différents.

Ces interprétations sont fondées sur la comparaison entre la situation réelle et ce qui devrait normalement se passer dans de telles circonstances. Par le décalage entre la situation réelle et ce "stéréotype", elle fait des inférences très variées sur les caractères et les plans des personnages.

Les défauts de ce système sont la classification trop stricte des différents niveaux d'interprétation, ainsi que la définition trop détaillée et limitée des situations stéréotypées.

Par contre, cette étude a l'avantage de bien mettre en évidence l'importance de tout ce qui reste implicite dans un dialogue : pour communiquer, on ne se limite pas à interpréter l'explicite, mais aussi tout ce qui reste non-dit, volontairement ou non.

Les études menées sur ce sujet ne sont pas le seul fait des informaticiens. Les travaux faits par les sémanticiens, comme Ducrot (Ducrot 72), montrent l'intérêt que ce domaine suscite actuellement.

Nous allons évoquer maintenant des travaux sur la conversation humaine qui ont été menés en linguistique ou en psycho-linguistique. Nous avons essayé d'en dégager des principes pour gérer les dialogues auxquels notre système devra prendre part.

I.2- Le dialogue, phénomène social.

Quand on examine des conversations tenues entre des personnes, le manque de formalisme apparent semble évident. Toutefois, il est facile de réunir un large consensus sur la validité d'une intervention au cours d'une conversation, située dans un cadre particulier, entre certaines personnes, sinon les conversations n'existeraient pas ! Mais il est difficile de cerner les raisons qui nous font préférer une formulation ou un enchaînement acceptable.

Les travaux de Grice (Grice 75) sur ce domaine sont intéressants. Il a mis en évidence des règles simples portant à la fois sur le fond et sur la forme des interventions prononcées dans une conversation. Il a montré que ces règles sont suffisamment strictes et universelles : pour transmettre un message on respecte généralement ces règles; si on les viole en apparence, c'est alors cet écart lui-même qui est chargé de sens.

A la base de ces règles, on trouve le "principe de coopération" qui peut se résumer ainsi : toute contribution verbale à une conversation doit être telle que l'impose l'état de la conversation au moment où cette contribution est accomplie.

Pour détailler ce principe, Grice énonce quatre maximes concernant les propriétés d'une intervention portant respectivement sur la Quantité, la Qualité, la Pertinence et la Manière. En d'autres termes, pour bien se faire comprendre, il faut apporter sa contribution au moment opportun, en étant suffisamment explicite, sans donner trop de détails superflus. La violation d'une de ces règles amène un interlocuteur à déduire des implications que l'on peut qualifier de conversationnelles, puisqu'elles ne sont pas contenues dans le sens des mots proposés.

Par exemple :

A dit "Je n'ai plus de cigarettes"

B répond "Il y a un café à cent mètres à droite"

A doit comprendre que ce café est ouvert, et qu'il fait office de bureau de tabac, et enfin que B ne fume pas, ou n'a plus de cigarettes, ou n'est pas prêt à lui en offrir une.

Il faut noter à ce sujet que chaque interlocuteur sait que cette interprétation aura lieu, il doit donc en maîtriser les causes. Un système, pour tenir une place crédible dans un dialogue, devra donc respecter les règles énoncées; sinon il risque de provoquer chez l'utilisateur une interprétation erronée de ses énonciations. On rejoint alors les travaux menés par certains linguistes comme A. Borillo (Borillo 77) qui ont montré l'importance de la prise en compte de la situation réelle d'énonciation pour comprendre et participer à un dialogue.

Des philosophes anglo-saxons se sont aussi récemment intéressés à ce qu'il est convenu d'appeler des "actes de paroles". En particulier, à la suite des travaux réalisés par Austin (Austin 62), puis Searle (Searle 79) sur la portée locutoire (le fait de "dire quelque chose"), illocutoire (un acte "effectué en disant quelque chose") ou perlocutoire (un acte "réalisé par le fait de dire quelque chose") des actes de langages. C'est à la suite de ces travaux qu'une parole ou une intervention sont devenues des actes à part entière, faisant partie de plans des personnages qui les prononcent. Ils permettent directement ou non de parvenir au but fixé.

C'est sur cette partie de dialogue que sont fondés les travaux d'Allen, Perrault et Cohen (Allen 80) (Perrault 78). Ils cherchent à reconnaître le plan de l'interlocuteur lorsqu'il s'adresse à la machine, afin de lui fournir une réponse qui corresponde aux buts qu'elle a su identifier, même si ce plan n'est pas explicitement exprimé.

En conclusion, les résultats que nous avons retenu pour les mettre en oeuvre dans les procédures que nous avons réalisées portent sur les points suivants :

- le dialogue est un phénomène social qui a ses règles propres, portant aussi bien sur le fond que sur la forme;

- il se déroule dans un environnement qui influe sur le comportement des personnes.

I.3- Le dialogue dans un système de questions-réponses.

Notre but n'est évidemment pas de traiter n'importe quel type de dialogue. Les études précédentes ont montré à quel point ce phénomène est complexe.

Le dialogue auquel nous nous intéressons intervient dans un système de questions-réponses qui n'est pas destiné particulièrement à des spécialistes mais plutôt à une utilisation grand public. De ce fait, il pourra être adapté à d'autres applications : un public de spécialistes demande une interface aussi soignée que Monsieur tout le monde. La différence essentielle se trouve plutôt dans le vocabulaire utilisé, et ne concerne donc pas les procédures de dialogue. On s'interdit aussi d'utiliser des interfaces "ad hoc" pour un système particulier (comme dans ONCOCIN (Langlotz 83)).

Le fait de se situer dans un système de questions-réponses fixe les rôles de chacun des interlocuteurs dans le dialogue qui va s'établir.

L'utilisateur va se présenter avec un problème qu'il souhaite résoudre en effectuant des recherches dans une base de connaissances. Le plan de l'utilisateur potentiel est donc de trouver des informations, ce qui peut être à l'origine d'une ou de plusieurs interrogations, afin de résoudre son problème. Les systèmes auxquels nous nous intéressons ont pour plan de toujours essayer de trouver les renseignements qui leur sont demandés, quitte à faire des approximations quand une solution n'est pas immédiate. C'est d'ailleurs l'utilisation de ces approximations qui rend le dialogue nécessaire et

intéressant; ce sont elles qui provoquent des interventions aussi bien de la part de l'utilisateur, que de la machine comme nous le verrons plus loin.

Ce que nous avons voulu faire, c'est mettre en place des procédures qui permettent de gérer un dialogue complet, et non pas une simple session qui comprenne une seule interrogation.

Ces procédures concernent uniquement les techniques de dialogue. Nous ne nous intéresserons ni au problème de génération, ni à celui de l'analyse. Nous travaillons à partir des représentations internes. Celles que nous utilisons sont fondées sur une grammaire de cas (Fillmore 68), mais les programmes développés peuvent s'adapter aisément à toute autre formalisme.

Avant que n'interviennent les procédures que nous avons mises au point, un analyseur prend en compte les interventions d'un utilisateur et les traduit en structure interne. La représentation interne des phrases est analogue à celle qui existe dans la thèse de Sabah (Sabah 76), et est créée par l'analyseur de Rady (Rady 83). Nous reviendrons dans le deuxième chapitre sur les détails de cette représentation.

Lorsqu'un mot ou une structure lexicale est inconnu, l'analyseur se charge de le signaler à l'interlocuteur.

Pour le travail que nous présentons ici, nous avons considéré qu'étaient résolus les problèmes de référents, d'anaphores et d'ellipses. Nous savons que ceci est un peu artificiel dans la mesure où ces problèmes ne peuvent pas toujours être complètement pris en charge par un analyseur, et que justement, c'est par un dialogue qu'un certain nombre d'ambiguïtés pourraient être levées.

Toutefois, nous avons pensé que ce problème était suffisamment indépendant de la gestion d'un dialogue concernant une recherche dans une base de connaissances pour les dissocier dans une première approche. De même, nous n'essaierons pas de mettre en place des procédures de génération. Elles posent des problèmes spécifiques comme les questions d'ordonnement des propositions (voir Blah (Weiner 80)), et des problèmes "symétriques" de ceux qui existent en analyse, comme l'utilisation des pronoms. Cependant nous avons déterminé les éléments à exprimer, c'est à dire la "quantité" de choses à dire (comme la définit (Grice 75)).

Nous n'avons pas abordé non plus les questions d'apprentissage, ni sur le vocabulaire, ni sur la base de connaissances. Toutefois, nous reviendrons plus loin sur les idées que nous avons à ce sujet, et sur les capacités potentielles que donnent les procédures mises en place dans ce travail.

Ces limites étant précisées, nous allons présenter les techniques que nous avons élaborées pour gérer un dialogue.

Le dialogue ne sera pas guidé par la machine, l'initiative reste à l'utilisateur, qui introduit le premier sujet, et a ensuite la liberté d'en changer, ou de revenir en arrière. Il nous faut donc suivre les thèmes qui sont abordés au cours du dialogue.

1.3.1- Les thèmes développés au cours du dialogue.

Nous introduisons ici la notion de "thème" ou de "sujet". Précisons le sens que nous lui donnons dans ce travail.

Dans un système de questions-réponses, nous considérons un thème comme étant l'objet d'une demande provoquant une recherche dans la base de connaissances pragmatiques. Nous ne considérons pas le thème comme une notion définie en opposition avec celle de rhème, comme le font les linguistes. En effet, cette définition est

donnée en linguistique de phrase, et dans notre travail sur le dialogue, nous avons repris une définition plus large, qui se rapproche de celle de sujet d'un discours.

Dans l'application sur laquelle nous avons travaillé, la base de connaissances concerne des professionnels des pages jaunes de l'annuaire. Dans ce cadre, un thème peut être simplement un nom de professionnel; sur une demande telle que :

"Je voudrais le numéro d'un garagiste",

le thème est interprété comme "être garagiste"; mais plus souvent, il s'agit d'un couple prédicat-objet. Dans ce cas, le prédicat est l'action à effectuer sur l'objet, et la recherche porte sur le professionnel susceptible d'effectuer cette action. Si la demande de l'utilisateur est :

"Je cherche quelqu'un qui puisse construire un
garage dans mon jardin",

le thème reconnu est "construire un garage".

Nous utilisons alors cette notion comme la définit B.Grau (Grau 83); le premier cas, où le thème se réduit à un concept, correspond à une définition réduite du thème, possible par la connaissance du domaine d'application. Il n'y a pas d'ambiguïté dans cette définition dans la mesure où on est sûr qu'il ne s'agit pas de quelqu'un qui veut devenir garagiste.

Lorsque le thème isolé par le système correspond à un concept qui n'est pas reconnu comme pouvant se rattacher au concept professionnel, on considère que ce thème est incomplet et une procédure particulière devra se charger de demander à l'interlocuteur de le préciser, en lui proposant un choix de possibilités.

Nous avons donc défini le thème d'une façon plus précise que Grosz (Grosz 77) ou Sidner (Sidner 83), dans la mesure où notre domaine d'application est beaucoup plus ouvert que dans leurs travaux. Ainsi, on ne peut se contenter comme thème de "garage" dans le deuxième exemple présenté. Nous verrons plus loin en quoi cette définition nous est utile dans la détermination des procédures à appliquer.

Nous avons présenté ici la notion de thème telle qu'elle est utilisée dans l'application sur les pages jaunes de l'annuaire. La détermination du thème a constitué une partie du travail de B. Grau (Grau 83), et nous nous contentons d'en reprendre les résultats. Soulignons toutefois que les procédures que nous mettons en place dans ce travail peuvent être utilisées dans d'autres applications, dans les limites fixées précédemment (§1.3).

1.3.2 Le déroulement du dialogue.

Nous allons voir quel est le schéma général que suit un dialogue. L'utilisateur pose une première question, à laquelle le système doit répondre. Pour cela une recherche est entreprise dans la base de connaissances. Si elle aboutit, une réponse est donnée; sinon une procédure de "rattrapage" fait en sorte de poser une question pertinente à l'utilisateur.

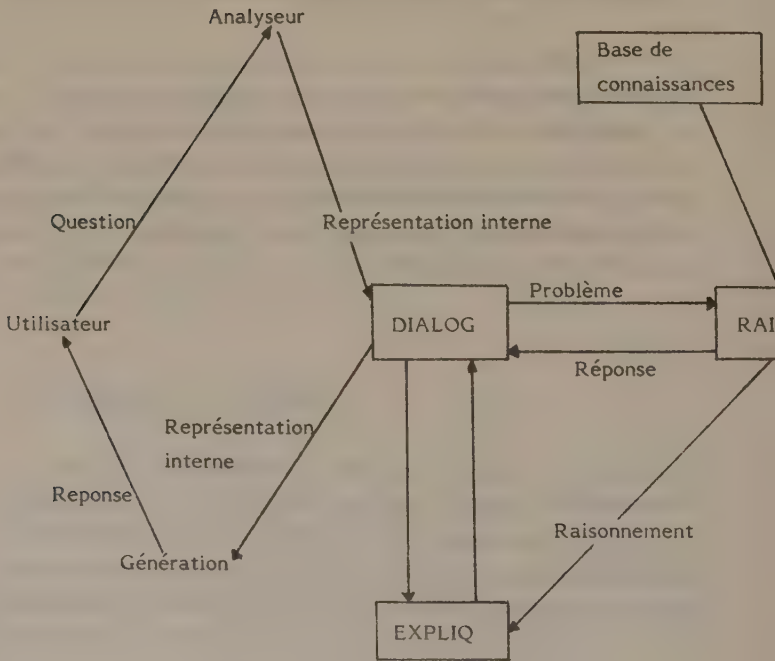
Par pertinente, nous entendons que la réponse faite par l'utilisateur devrait fournir les éléments nécessaires au déblocage de la recherche. L'utilisateur peut alors répondre à la question si le système lui en a posé une, ou demander des explications, sur ce qu'on lui a dit. Il peut également passer à un autre sujet, quitte à revenir en arrière par la suite.

L'attitude qu'un utilisateur potentiel attend de la machine est une attitude coopérative, c'est à dire réellement informative, comme

l'affirment B. Webber et al. (Webber 83). Il faut donc pouvoir répondre à une demande d'explications en localisant le motif de l'incompréhension qui a provoqué la demande. On peut alors formuler une explication de façon claire et cohérente. Il ne s'agit nullement de fournir tous les détails de la recherche dans la base de connaissances, mais d'explicitier les points difficiles.

Webber insiste également sur ce qui rend une justification claire et compréhensible. Il faut pour cela être succinct : adopter ce que Sadock appelle le "modus brevis" (Sadock 77). Toutefois, il faut rester suffisamment explicite. On doit déterminer un "niveau" de justification, tel qu'en dire plus deviendrait lassant et peu informatif, mais en dire moins rendrait l'explication obscure. On retrouve ici les principes de Grice sur la quantité.

Nous pouvons représenter ainsi le système sur lequel nous avons travaillé



DIALOG : module chargé de la gestion du dialogue

RAISON : module chargé de la recherche dans la base de connaissances

EXPLIQ : module chargé de justifier et d'expliquer les actions de RAISON

Fig. 1.1 Représentation du système

Reprenons le cheminement dans ce schéma.

Tout dialogue est initialisé par une question de l'utilisateur. Par la suite, ses interventions se diversifient : réponse, demande d'explications, ou nouvelle question. Toute intervention est transmise à l'analyseur qui en construit la représentation interne, et en détermine le thème. Ces deux éléments sont fournis au module DIALOG. Ce module va chercher à reconnaître le thème qui lui est fourni parmi ceux qui ont déjà été abordés.

Deux cas peuvent alors se produire :

1) l'intervention est une question sur un thème nouveau.

C'est une question car un changement de thème, tout comme l'initialisation du dialogue, ne peut être qu'une question concernant la base de connaissances. Celle-ci est alors transmise au module RAISON.

2) l'intervention concerne un thème déjà abordé.

Si celle-ci est une interrogation, il s'agit d'une demande d'explication ou de justification sur une réponse fournie par le système plus tôt dans le dialogue. Elle est alors transmise à EXPLIQ.

Si l'intervention est une information, il s'agit d'une réponse à une question posée par le système en vue d'aider RAISON. Elle est donc utilisée afin de trouver une réponse définitive.

RAISON doit chercher une réponse au problème qui lui est transmis. Ce module n'est pas concerné par les procédures de dialogue. Il est indépendant de DIALOG et de EXPLIQ : ce sont des modules qui interagissent. De ce fait, son fonctionnement ne nous intéresse pas ici a priori.

Il en est de même pour le mode de représentation choisi pour établir la base de connaissances pragmatiques. Ce peut être tout aussi bien un système-expert travaillant sur un corpus de règles, qu'un système fondé sur l'utilisation de stéréotypes, ou de toute autre représentation. Nous verrons d'ailleurs que le système sur lequel nous avons réalisé une application pratique appartient encore à une autre catégorie (Sabah 83).

Toutefois, nous avons besoin que RAISON constitue, au fur et à mesure de son déroulement, une trace du raisonnement qu'il est en train de mener, cette donnée devant être transmise ensuite à EXPLIQ. Cette trace est construite automatiquement et indépendamment du mode de raisonnement. Ceci est rendu possible par l'application d'une procédure préalable à RAISON. Celle-ci a pour but de lui ajouter automatiquement les instructions qui provoquent la constitution de la trace lorsque RAISON est mis en œuvre.

Dans le chapitre suivant, nous verrons en détail comment ceci est réalisé.

Deux hypothèses sont à envisager à la fin du travail de RAISON:

- la recherche dans la base de connaissances a pu être menée à bien ; il transmet alors les éléments de la réponse à DIALOG qui se contente de la fournir au module de génération. Elle sera ensuite présentée à l'utilisateur.

- la recherche n'a pu aboutir complètement. Ce constat d'échec est transmis à DIALOG. Celui-ci va alors faire appel au module EXPLIQ qui a reçu la trace du raisonnement ébauché par RAISON.

Il faut alors mettre en place une procédure qui permette de construire la représentation interne d'une question à poser à l'utilisateur, afin de débloquer le raisonnement.

1.3.2.1 - La relance du dialogue.

Il nous semble en effet important de ne plus donner à l'utilisateur des réponses qu'il ne peut, à juste titre, considérer comme coopératives, du type :

"Je ne peux vous répondre. Posez votre question différemment."
ou encore :

"Pouvez-vous détailler votre requête ?".

Sabatier soulève également ce problème au sujet des messages formulés par un analyseur (Sabatier 83). Placé devant ce genre de réaction, l'utilisateur ne sait que faire et, souvent, abandonne. Cette attitude est normale, dans la mesure où le système ne lui ayant apporté aucun élément nouveau dans cette intervention, il ne possède aucun moyen de savoir en quoi sa demande pouvait se révéler inacceptable, ou seulement difficile à résoudre. C'est une rupture du principe de coopération, qui provoque celle du dialogue, à moins que l'utilisateur ne tienne vraiment à obtenir un résultat. Dans ce cas un problème se pose à lui : comment reprendre le dialogue.

De son point de vue, mis à part des cas très rares, il a exposé son problème clairement, et même s'il essaie à nouveau de l'exprimer, il n'est pas évident qu'il puisse obtenir une réponse. S'il ignore quelle est l'information manquante, il risque de ne pas la fournir dans une seconde demande (elle peut lui sembler tout simplement tellement évidente, qu'il ne pense pas à la donner).

Prenons un exemple, où le manque de détail dans la relance conduit à un échec :

Utilisateur : "Je voudrais un restaurant d'extrême-orient".

Système : "Précisez"

Le système connaît des restaurants chinois, japonais, vietnamiens et ne peut choisir entre ceux-ci, ni les fournir tous, la liste en serait trop longue.

Utilisateur : "Je voudrais manger des nids d'hirondelle"

Système : "Je ne connais pas ce terme"

On voit ici un blocage du dialogue, alors qu'une attitude plus coopérative du système aurait pu l'éviter. L'utilisateur ne connaissant pas le problème rencontré par le système, fournit une information qui ne permet pas sa résolution, au contraire il provoque un blocage complet.

A la place de cette attitude, le système doit essayer de déterminer la cause de l'échec de RAISON, et poser une question pertinente. Dans l'exemple précédent, proposer le choix entre les trois types de restaurant déterminés aurait permis de prolonger le dialogue beaucoup plus facilement.

Il faut pour cela se servir de l'ébauche de raisonnement qui a été conduit. Avec celle-ci, on peut voir à quel endroit le système a été bloqué. Souvent les causes de blocage sont dues à un manque d'information, ou de précision dans la requête. La plupart des systèmes ont pour but de trouver une réponse, et, quitte à effectuer des approximations, ils en obtiennent généralement trop. Il faut alors choisir dans cet ensemble un critère qui soit suffisamment sélectif pour permettre de réduire le nombre de possibilités.

Le travail de DIALOG est donc d'examiner l'ensemble des éléments que RAISON a réunis, et de retrouver, dans les connaissances qu'il peut avoir au sujet de ces éléments, des caractères distinctifs.

Lorsque DIALOG a réussi à déterminer les éléments constituant la question à poser, il la transmet au module de génération qui va la soumettre à l'utilisateur.

I.3.2.2- L'explication.

Revenons maintenant au cas où EXPLIQ doit répondre à une demande d'explication. Nous avons vu que, dans ce cas, DIALOG a reconnu que l'intervention était une interrogation concernant un thème déjà abordé. EXPLIQ doit alors retrouver à son tour le travail réalisé par RAISON à ce sujet.

Il retrouve la trace du raisonnement qui avait été mené lorsque l'utilisateur avait abordé ce thème. En comparant cette trace avec la question posée, il peut alors déterminer sur quel point du raisonnement porte la demande d'explication. En effet, la trace nous permet de savoir quelles actions a accomplies RAISON, ainsi que le résultat qu'il a obtenu.

La demande de l'utilisateur peut porter soit sur le résultat complet, soit sur une partie du raisonnement. En fonction de ce critère, EXPLIQ choisira les éléments de la trace à retenir pour fournir à l'utilisateur une explication satisfaisante. Dans les deux cas, ce choix consiste à éliminer les éléments inutiles de la trace, soit parce que ce sont des détails, qui ne sont pas informatifs car trop précis, soit parce qu'ils ne concernent pas l'interrogation en cours.

Les éléments sont repris par DIALOG qui les transmet au module de génération, afin de fournir à l'utilisateur la réponse qu'il attend.

Nous avons donc vu quelles étaient les différentes procédures mises en place pour gérer un dialogue dans un système de questions-réponses.

Dans le travail que nous avons réalisé, nous avons écrit les modules DIALOG et EXPLIQ, ainsi que la procédure qui permet à RAISON de transmettre une trace de son action à EXPLIQ.

Nous allons montrer sur trois exemples comment ces procédures sont mises en œuvre.

1.3.3- Exemples.

Voyons un premier exemple où deux thèmes différents sont abordés et imbriqués.

Exemple 1 :

U1 : Je voudrais l'adresse d'un restaurant chinois ouvert le soir.

S1 : Dans quel quartier souhaitez-vous aller?

U2 : Je voudrais d'abord savoir si vous connaissez un numéro de téléphone de taxi de nuit?

S2 : Composez le numéro de la compagnie TRUC ou MACHIN.

U3 : Donnez-moi un restaurant dans le XVI ème arrondissement.

S3 : Vous avez le Mandarin N°-----

ou le Jardin du ciel N°-----

(Pour des raisons de présentation, nous montrons ici des exemples où le système fournit le numéro de téléphone recherché, mais nous verrons que notre système ne peut pas toujours descendre jusqu'à ce niveau de précision).

Dans cet exemple, nous pouvons analyser les interventions de l'utilisateur de façon simple : il souhaite sortir dîner dans un restaurant chinois, et ne dispose pas de moyen de transport.

Il commence par s'enquérir d'un restaurant ouvert la nuit. La demande est transmise à RAISON. Celui-ci se trouve dans un cas où il ne peut fournir de réponse précise : le nombre de restaurants chinois ouverts le soir à Paris est trop important. Il transmet la liste à EXPLIQ qui, en liaison avec DIALOG, va déterminer un critère distinctif entre ces restaurants.

Nous reviendrons plus loin sur la méthode adoptée par ces deux modules pour trouver un critère discriminatoire, en fonction de connaissances pragmatiques sur les éléments composant la liste fournie par RAISON.

A l'aide de ce critère, DIALOG décide de la structure de la question à poser et la transmet au module de génération (S1).

L'interlocuteur laisse cette question en suspens pour évoquer la deuxième partie de son plan : le moyen de transport (U2).

Le système, placé devant ce changement de thème (la recherche d'un taxi n'a rien à voir a priori avec celle d'un restaurant chinois) fait appel à RAISON pour cette dernière demande. Celui-ci répond sans difficulté : il n'y a que deux solutions possibles, il les propose donc toutes les deux (S2).

L'interlocuteur revient alors au premier thème évoqué en répondant à la question qui précédait (U3).

Le système reconnaît ce retour et sait qu'il est de nouveau question d'un restaurant chinois ouvert la nuit. Avec le complément d'information qui lui est fourni, il peut alors donner une réponse.

Voici deux autres exemples de dialogue, qui illustrent les procédures de justification à l'intérieur d'un même thème.

Exemple 2 :

U1 : Je cherche à réparer mon auto-radio

S1 : Je vous propose un garagiste ou un réparateur d'appareils électriques.

U2 : Pourquoi me proposez-vous un garagiste ?

S2 : Parce que votre auto-radio fait partie de votre automobile.

U3 : D'accord. Je voudrais l'adresse d'un garagiste.

S3 : Dans quel quartier ?

U4 : Dans le 5ème arrondissement.

S4 : Vous pouvez appeler le garage REPARTOU au 336-25-25.

Dans cet exemple, l'utilisateur présente sa demande (U1). Elle est transmise à DIALOG, puis à RAISON. Ce dernier trouve une solution partielle au problème qui lui est posé, il ne peut décider entre les deux professions susceptibles de répondre à la demande : ce choix est fonction d'un critère qu'il ne maîtrise pas. Il demande alors à l'utilisateur de faire un choix (S1).

L'utilisateur demande une justification (U2) : il ne comprend pas pourquoi on lui propose un garagiste. DIALOG reconnaît cette demande et la transmet à EXPLIQ. Celui-ci reprend la trace du raisonnement non complètement abouti que lui a transmis RAISON à la suite de sa recherche sur ce thème. Il est alors en mesure de retrouver le lien entre garagiste et auto-radio.

Dans cette réponse (S2), le système limite son explication au point souligné par l'utilisateur. Il est inutile de préciser toutes les qualités d'un auto-radio (pourquoi il est assimilé à un appareil électrique, par exemple). Ce n'est pas sur cette partie du raisonnement que l'utilisateur bute. Il est également inutile de préciser qu'un garagiste répare les automobiles : ce type de renseignement est trop évident pour rester informatif.

L'utilisateur, satisfait par cette justification fait son choix (U3). RAISON est alors relancé avec cet élément nouveau, mais il ne peut aboutir tout à fait : on retrouve la situation de l'exemple précédent, le système ne trouve pas de solution simple et demande de choisir un quartier (S3). L'utilisateur répond et on aboutit à une réponse satisfaisante (U4 et S4).

Exemple 3 :

U1 : Je voudrais me débarrasser des cafards qui envahissent
ma maison.

S1 : Je vous conseille de vous adresser à un dératiseur.

U2 : Pourquoi ?

S2 : Parce que les dératiseurs débarrassent les habitations
des animaux nuisibles.

U3 : D'accord. Donnez-moi un numéro.

S3 : Appelez TUTOUT au 362-12-21.

Dans cet exemple, un autre sujet est introduit par l'utilisateur : se débarrasser des cafards. Le système propose de s'adresser à une entreprise de dératisation. L'utilisateur ne voit alors aucun lien entre les deux éléments et demande une justification (un cafard n'est pas un rat) sur l'ensemble du résultat qui lui est présenté.

DIALOG, en liaison avec EXPLIQ, reprend alors la trace fournie par RAISON. Il justifie sa réponse par le lien entre cafard et animal nuisible, joint au fait que les dératisations concernent tous les animaux nuisibles.

Nous avons vu dans ces deux derniers exemples, le passage du thème "se débarrasser des cafards" à celui de "se débarrasser des animaux nuisibles", ou de "réparer un auto-radio" à "être garagiste", par exemple. La trace du passage, introduit par le système, est conservée, mais on note qu'il s'agit toujours d'un même thème principal : ce n'est pas un changement complet de thème, mais une déviation (voir (Grau 83)), tout ceci est relié à une même préoccupation de l'utilisateur. Par contre, dans le premier exemple, le passage de "chercher un restaurant" à "chercher un taxi", est considéré comme un changement : il n'y a pas a priori, pour le système, de lien direct entre ces deux thèmes. Ils ne sont reliés que dans le plan de l'utilisateur.

Dans ce chapitre, nous avons vu les principes qui nous ont permis de définir les grandes lignes de notre travail, et nous les avons illustrés sur trois exemples. Nous allons montrer à présent quelles sont les représentations que nous avons mises en place pour atteindre les buts que nous nous sommes fixés.

- CHAPITRE II -

Les Représentations

Dans ce chapitre, nous allons présenter les différentes représentations que nous avons mises en place pour gérer les procédures de dialogue évoquées dans le chapitre précédent.

Nous commencerons par présenter la structure des connaissances générales concernant la syntaxe, la sémantique et la pragmatique. Nous reprendrons ici des représentations mises en place dans notre équipe, et qui ont déjà fait l'objet de diverses publications (Sabah 83), (Rady 83), (Grau 83).

Nous évoquerons ensuite le mode de représentation des connaissances particulières à l'application sur laquelle nous avons travaillé. Les notions présentées alors ne sont pas directement liées aux procédures de dialogue, mais elles permettront d'éclairer l'exposé des exemples que nous donnons.

Enfin, nous présenterons les représentations plus spécifiques des connaissances nécessaires pour gérer un dialogue, à savoir celles des deux interlocuteurs (le système et l'utilisateur) ainsi que celles qui permettent le suivi du dialogue.

II.1 - Représentation interne des phrases

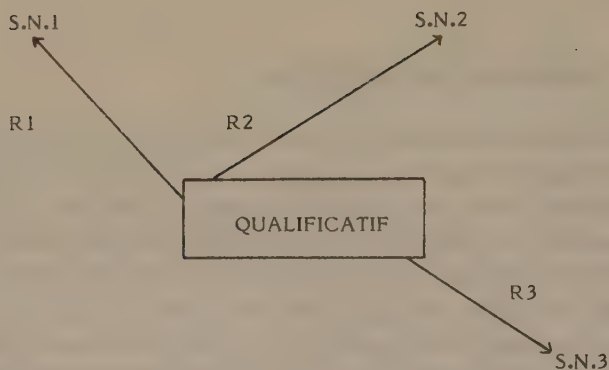
La représentation interne choisie est une structure relationnelle construite sur le modèle des grammaires de cas, modèle introduit par (Fillmore 68). Dans cette théorie, une phrase est considérée comme un ensemble de syntagmes nominaux, chacun d'eux étant relié au qualificatif de la phrase (généralement le verbe, mais aussi éventuellement un adjectif, une préposition ou un nom). Chaque liaison exprime la nature sémantique du lien qui existe entre ce qualificatif et le syntagme nominal en question. Cette structure est également en accord avec (Tesnière 53) pour qui le verbe est au centre de la phrase.

Suivant approximativement (Simmons 73), nous avons choisi les six cas ci-dessous :

- . SOURCE
- . AGENT PRINCIPAL
- . THEME
- . INSTRUMENT
- . BUT
- . LIEU

Sans entrer dans les détails, disons rapidement que la source exprime l'origine d'un processus, l'agent l'auteur d'une action et le thème ce sur quoi porte cette action ou ce processus. L'instrument aide l'agent à effectuer cette action, au profit d'un certain but.

La phrase peut donc se représenter de la façon suivante :



Chaque syntagme (S.N.i) est lié au qualificatif par une relation sémantique (Ri).

Remarquons cependant que ce type de représentation ne rend pas compte de toute l'information présente dans la structure de surface. En effet, si la structure que nous venons de détailler nous permet d'avoir des représentations internes semblables pour des phrases qui ont sensiblement le même sens, des différences purement syntaxiques peuvent apporter des nuances non négligeables.

La forme active et la forme passive d'une même phrase, par exemple, si elles ont le même sens, ne mettent pas l'accent sur le même objet.

Ainsi, pour tenir compte de ces faits, nous inclurons la structure casuelle de la phrase dans une structure récursive mettant en évidence l'objet principal de la pensée et le prédicat qu'on lui applique, dans un certain contexte.

La structure que nous avons adoptée peut être représentée par la figure suivante :

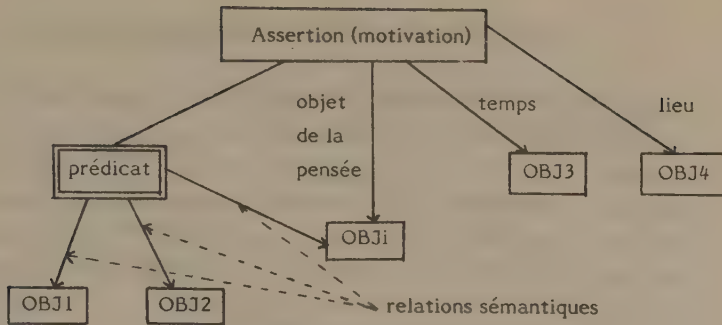


Fig II.1 - Représentation interne d'une phrase

La motivation contient les informations générales relatives à la phrase entière. Elle nous permet de savoir s'il s'agit d'une information (motivation affirmative, négative ou restreinte), ou d'une demande (motivation question polaire, question objet ou question causale).

Le prédicat contient le qualificatif de la phrase, éventuellement un adverbe venant le modifier, et des relations sémantiques qui le lient à des objets. Ces objets sont des occurrences de concepts qui figurent dans le réseau sémantique (§II.2.1.2).

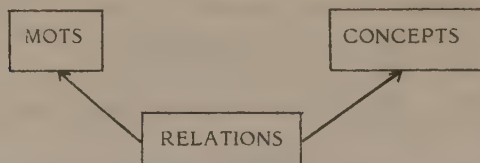
Chacun des objets peut être précisé par une sous-phrase. La motivation nous donne alors la nature du lien entre les deux phrases.

II.2 - La représentation des connaissances

Les systèmes de traitement du langage naturel attachent trois types d'informations à chaque entrée lexicale : les informations morphologiques, syntaxiques, sémantiques. Le système doit également avoir des connaissances propres au domaine traité : les informations pragmatiques.

II.2.1 - Représentation des connaissances syntaxico-sémantiques

Le système de représentation des informations syntaxiques et sémantiques peut être schématisé ainsi :



Pour expliquer la représentation que nous avons choisie, prenons un exemple : le mot 'achat'. Syntaxiquement, il n'est pas ambigu : c'est un nom; il a cependant deux sens : l'action d'acheter ou ce que l'on a acheté.

Deux solutions sont classiquement utilisées pour tenir compte de ce fait. On peut avoir deux entrées distinctes dans le dictionnaire, ayant les mêmes informations syntaxiques mais des informations sémantiques différentes.

La deuxième solution (cf. grammaire transformationnelle) consiste à considérer 'achat' comme un mot dérivé, le lexique ne contenant qu'un seul mot 'acheter'. L'ambiguïté vient du fait qu'il existe plusieurs dérivations possibles.

Nous avons préféré une troisième solution, qui se rapproche de la distinction établie par (Halliday 61) entre "item grammatical" et "item lexical". Nous dirons que 'achat' et 'acheter' sont deux MOTS différents qui renvoient au même CONCEPT 'l'évènement d'acheter'. On résout ainsi aisément les cas de polysémie (un mot peut correspondre à plusieurs concepts) ou de synonymie et homographie (plusieurs mots sont liés au même concept).

II.2.1.1 - Les connaissances morpho-syntaxiques

Les informations morphologiques permettent une certaine économie dans la représentation du lexique : chaque entrée comprend un "mot-tête" et un ensemble de règles qui permettent de retrouver toutes les formes du mot ou d'identifier, à partir d'une forme donnée, le "mot-tête" correspondant. L'analyseur morphologique (repris de (Pitrat 79)) repose sur la combinaison de racines de mots avec des suites de terminaisons : on indique, sous forme déclarative, au moyen de "matrices de conjugaisons", quelle racine se combine avec quelle terminaison pour produire une forme d'un mot.

Les informations syntaxiques se rapportent essentiellement à la catégorisation des mots de la langue, afin de déterminer les combinaisons possibles, formant des énoncés acceptables. Elles sont simplement organisées sous forme de liste de traits,

Cet ensemble d'informations forme le fichier VOCABULAIRE (ou dictionnaire) représentant les MOTS.

II.2.1.2 - Représentation des CONCEPTS

Deux modèles sont généralement employés pour représenter les informations sémantiques :

- . le modèle componentiel où l'on décrit chaque élément par la présence ou l'absence d'un certain nombre de traits sémantiques.
- . le modèle associatif, ou réseau sémantique, où un ensemble de relations relient les concepts les uns aux autres, décrivant ainsi leur sens.

Nous avons choisi la seconde solution. La représentation est alors construite autour des trois idées principales suivantes :

- Une structure de graphe des concepts, dans laquelle se réalise une transmission "par héritage" des propriétés d'un concept à ses composants.
- La représentation de ces propriétés par une liste de "liens sémantiques" qui relient le concept à d'autres.
- La "transitivité" de certains liens, ce qui évite d'avoir à représenter de nombreuses relations.

Un premier type de lien ("sorte-de") permet de représenter le graphe des concepts qui traduit leur structure hiérarchisée en ensembles et sous-ensembles. Les autres liens précisent les traits (ou "qualités") attachés à chaque concept.

Donnons une partie du graphe comme exemple :

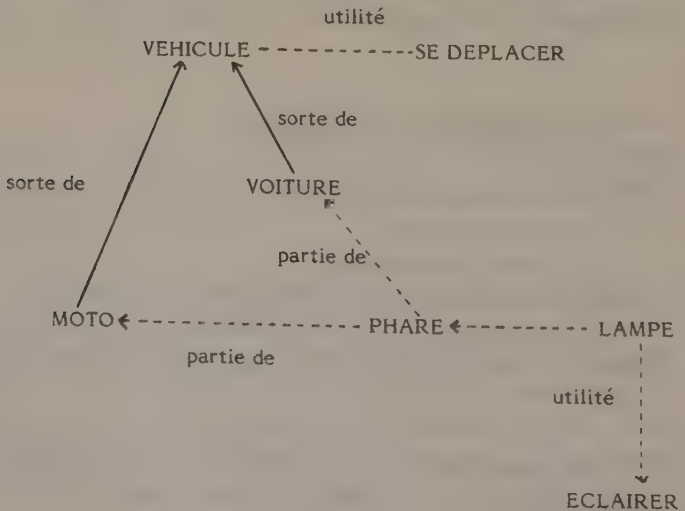


Fig II.2 Extrait du graphe des concepts

Ce graphe exprime les faits suivants :

Moto et voiture sont deux espèces de véhicules. Un véhicule sert à se déplacer. Une voiture et une moto contiennent un phare qui contient une lampe qui sert à éclairer.

Dans la partie du vocabulaire qui correspond aux verbes, les mêmes traits sémantiques sont utilisés mais avec un sens un peu différent. En effet, le sens d'un verbe est plutôt défini par des qualités portant sur les noms qui peuvent lui être liés que par des traits sémantiques qui lui sont propres.

Ainsi le concept soigner sera-t-il précisé par les exigences qui portent sur l'agent : être humain (celui qui soigne), le thème : maladie (que l'on soigne); le but animé (celui qui est soigné) et l'instrument : médicament ou traitement (avec quoi on soigne).

Pour compléter cette représentation, nous avons introduit une notion de "transitivité" des liens sémantiques. Cette notion originale de calcul permet une économie de place importante.

En plus des informations que nous venons de présenter, nous avons établi une table de correspondance entre les divers types de liens qui apparaissent dans cette structure. Ceci nous permet de transmettre certaines qualités par "transitivité".

Cette table - du type "loi de composition interne" - indique par exemple :

Sorte-de o Sorte-de = Sorte-de
Partie-de o Sorte-de = Partie-de
Contenir o Utilité = Capacité

...

Ces relations expriment que si un élément est lié à un autre par un lien du premier type et si cet autre est lié à un troisième par un lien du deuxième type, un lien du type résultat existe entre le premier et le troisième élément.

Elle peut également montrer que deux relations ne peuvent être composées (comme par exemple matière et partie-de). Le système peut alors retrouver quand il en a besoin des informations implicites comme :

Une voiture peut éclairer
Une voiture sert à se déplacer
Une lampe est une partie de voiture
...

Les économies de mémoire réalisées par l'utilisation de cette représentation du vocabulaire sont donc de divers types :

- . un élément n'apparaît qu'une seule fois alors qu'il peut intervenir dans la description de plusieurs concepts.
- . de nombreuses relations entre les éléments ne sont pas exprimées : le programme les retrouve selon ses besoins.

II.2.1.3 - Les relations entre mots et concepts

La distinction mot-concept implique l'existence d'un ensemble de règles de correspondances qui, dans chaque contexte, déterminent les concepts appropriés liés aux mots de l'énoncé.

Dans la version actuelle de l'analyseur, nous avons jugé souhaitable de représenter les règles de correspondance de façon déclarative. Ainsi, en plus des deux fichiers 'MOT' et 'CONCEPT', un fichier 'CORRESPONDANCE' a été mis en place. Il permet alors de trouver le sens des constructions syntaxiques, en précisant le rôle sémantique de tel ou tel type de complément.

Donnons un exemple :

On indique une correspondance (corr.i) pour chaque structure syntaxique qui est en liaison avec un sens du concept correspondant.

Une structure syntaxique est représentée par un ensemble de rôles syntaxiques que peuvent avoir des mots liés au mot considéré; pour chacun d'eux, on précise s'il est obligatoire (obl.) ou non (fac.) ainsi qu'une éventuelle préposition qui l'introduit (écrit entre / / dans l'exemple ci-dessous) :

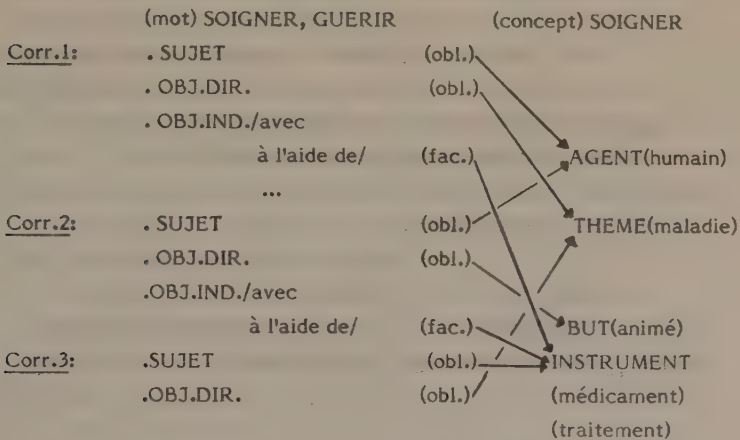


Fig II.3 Extrait du fichier des correspondances

La première correspondance indique la construction :

"Le médecin soigne la grippe (avec de l'aspirine)" avec médecin = agent, grippe = thème et l'éventuelle aspirine (facultatif) = instrument.

La deuxième correspond à : "Le médecin soigne Jean (avec de l'aspirine)" où Jean est ici le but. Quant à la troisième, il s'agit d'une forme "passive" : "L'aspirine soigne la grippe".

II.2.2 - Représentation des connaissances pragmatiques

L'application sur laquelle nous avons travaillé est l'interrogation d'un système remplaçant les pages jaunes de l'annuaire. Les informations qu'il nous faut représenter concernent donc les rubriques et les professionnels qui y figurent.

Nous définissons une rubrique de la façon suivante :

RUBRIQUE = NOM + DESCRIPTION

II.2.2.1 - Le nom

Il est destiné à représenter l'essentiel du sens de la rubrique. C'est un ensemble de quelques pointeurs vers les concepts, analogue au titre que l'on trouve actuellement dans les pages jaunes. Le nombre de concepts utiles nous semble être de 1 à 3.

Quelques exemples :

"auto-école"	: école, enseigner, conduite
"ambulance"	: ambulance, transporter, malade
"marchand de cycles"	: cycle, vente.

II.2.2.2 - La description

Un court texte permet ensuite de décrire de façon plus détaillée l'activité des professionnels inscrits sous cette rubrique. Ces phrases, données en français courant, sont conservées sous forme de structures internes produites par l'analyseur.

Ainsi, par exemple, "dans une auto-école, un moniteur enseigne la conduite d'une automobile". "On peut y prendre des cours de code sous forme audiovisuelle". "Certaines auto-écoles forment des moniteurs" etc...

II.3' - Représentation du raisonnement

Nous abordons ici les travaux que nous avons personnellement réalisés.

II.3.1 - Présentation générale

Beaucoup de systèmes s'intéressent aux interlocuteurs : savoir ce qu'ils sont, ce qu'ils pensent, ce qu'ils recherchent. Dans le cadre d'études portant sur les dialogues écrits comme celles menées par Schank et Lehnert (Schank 79), les auteurs se sont attachés à représenter les relations entre les différents interlocuteurs : qui ils sont l'un pour l'autre, quel devrait être leur comportement respectif dans une situation donnée,.

Dans le cadre de dialogue entre une machine et un utilisateur, Allen et Perrault (Allen 80) ont essayé d'avoir une représentation aussi précise que possible de l'utilisateur pour en déterminer les plans éventuels. Ce problème est très important, et nous verrons plus loin (§II.5) nos propositions pour le résoudre.

Mais la représentation qui reste la plus succincte, quand elle existe, est celle que la machine peut avoir de son propre rôle, et de la façon dont elle travaille.

Dans un dialogue humain, quand un interlocuteur (I1) réagit à une demande, il donne une réponse qui n'est, bien évidemment, pas entièrement justifiée et prouvée. Il présuppose chez son interlocuteur (I2) un certain nombre de connaissances qui lui permettront de reconstituer les étapes manquantes pour comprendre la réponse ou, au moins, en admettre la validité. Quand ce n'est pas le cas, il est naturel que (I2) poursuive son interrogation sur la dernière réponse fournie. Le refus d'apporter des éclaircissements de la part de (I1) serait alors une violation du principe de coopération sur lequel repose le dialogue (cf §I.2). On imagine mal quelqu'un dire qu'il n'a pas la moindre idée de la raison pour laquelle il a répondu ainsi, à moins d'être dans le monde de l'absurde.

Beaucoup de systèmes informatiques ont pourtant tendance à faire ce genre de réponse. Ils sont en mesure de mener à bien un travail qui peut être très complexe : recherche dans une base de connaissances avec des données plus ou moins bien définies,...; mais ces systèmes sont rarement capables de rendre compte de ce travail en l'expliquant et en le justifiant.

Les seuls systèmes ayant quelques compétences dans ce domaine sont les systèmes experts, ceci grâce à leur mode de fonctionnement. En effet, il leur suffit de fournir les connaissances utilisées (c'est-à-dire les règles activées) pour présenter une certaine forme de justification des résultats obtenus. Toutefois, ce n'est pas toujours satisfaisant (voir §1.1).

Dans le cas des systèmes dont le fonctionnement est procédural, ou à la fois procédural et déclaratif, ils ont besoin d'une représentation de ce qu'ils ont fait, pour expliquer leur démarche.

Pour cette raison, nous avons muni notre système d'une représentation du raisonnement qu'il mène chaque fois qu'il essaie de répondre à une demande de l'utilisateur. Il s'agit donc d'une représentation du travail réalisé par le module RAISON (voir fig 1.1) chaque fois qu'il est activé. Dans cette représentation, figureront à la fois les données prises en compte, la façon dont elles ont été retrouvées dans la base et leur utilisation. De plus cette représentation est construite automatiquement par le module RAISON lors de son déroulement .

Soulignons brièvement la double utilisation que nous souhaitons faire de cette représentation :

- déterminer ce qui constituera une réponse à une demande d'explication de l'utilisateur.
- déterminer ce qui permettra de poser une question à l'utilisateur en vue de faire aboutir un raisonnement inachevé (ou relancer le dialogue en cas d'échec).

Pour cela, il nous faut donc retrouver la démarche accomplie par le système quand il résout un problème donné. Ceci implique une représentation des actions que le module RAISON effectue à partir des données qu'il possède chaque fois qu'il est activé.

Précisons ce que nous entendons par action du programme.

Un système est conçu pour répondre à un certain nombre d'exigences concernant les cas qu'il est en mesure de traiter. Pour cela, un algorithme est élaboré et traduit dans un langage de programmation. Les actions qu'il faut représenter sont celles que le programme peut exécuter. Ce sont, en fait, les différents modules de cet algorithme.

La représentation du raisonnement mené lors d'une exécution de RAISON, correspond alors à l'activation d'une partie de l'algorithme : celle qui permet la résolution d'un cas particulier défini par les données de l'exécution en cours.

Pour réaliser la construction automatique d'une trace du travail de RAISON, nous avons choisi de procéder en deux étapes, afin de ne pas trop ralentir le fonctionnement de RAISON : l'un avant la mise en route du système, l'autre se déroule lors de chaque exécution.

La première étape consiste donc en la construction d'une représentation du programme RAISON : nous l'appellerons IMALGO. Celle-ci résume toutes les possibilités que le système est capable de prendre en compte et de quelles façons il les traite.

Au cours de chaque exécution, nous construisons une représentation (appelée IMEXEC) du raisonnement particulier qui est mené; elle correspond alors à l'extraction d'une partie de IMALGO dans laquelle les données prennent une valeur explicite.

En résumé, la construction de IMALGO a lieu avant la mise en oeuvre du système, et sa durée n'influe donc pas sur le temps de réponse du système; la construction d'une représentation IMEXEC a lieu lors de chaque activation du module RAISON.

II.3.2 - IMALGO : la représentation du module RAISON

Nous avons vu que la structure IMALGO représente l'ensemble des possibilités que le module RAISON peut prendre en compte.

Evoquons brièvement la façon dont cette structure est obtenue, (pour plus de détails, se reporter au ch. III).

La construction de IMALGO est réalisée automatiquement par une procédure qui examine le programme RAISON. Celle-ci recherche la structure du programme, l'enchaînement et l'imbrication des différents modules, et la représente sous forme arborescente.

Le choix de ce type de représentation nous est dicté par la nécessité de trouver une hiérarchie entre les actions du programme, comme nous allons le montrer plus loin (§II.5). De plus ce choix est cohérent avec la notion de programmation structurée dans laquelle des modules distincts interagissent. Le programme de traduction que nous avons réalisé est général et peut être appliqué à des programmes écrits dans des langages structurés tels que Lisp, Pascal (voir le travail réalisé à l'aide d'un système-expert dans (Revenu 82)), Simula... Cette traduction peut poser quelques problèmes avec des langages plus anciens (comme FORTRAN) où les règles de programmation sont beaucoup moins strictes : l'utilisation de branchements par GOTO venant perturber l'organisation du programme (voir (Laurent 82), (Adam 78) et (Fouet 82)).

Nous reviendrons plus en détail sur les problèmes liés à la construction de IMALGO dans le chapitre III.

Nous allons maintenant voir quelle est la structure d'IMALGO ainsi que les connaissances qu'elle contient.

II.3.2.1 La structure de IMALGO

Chaque noeud de la structure arborescente IMALGO représente une fonction du programme. Définissons ce que nous entendons par ce terme de fonction. Il s'agit ici d'un groupe d'instructions visant à aboutir à un certain but : réaliser une action donnée, tel que nous avons défini ce terme au §1.3.1. Une fonction du programme est donc la traduction d'une action, ou module de l'algorithme. Cette définition est plus large que celle qui existe dans un langage de programmation tel que Lisp. Une fonction peut coïncider avec une fonction Lisp, prédéfinie ou non, mais aussi avec un sous-ensemble isolé par le programmeur lors de l'écriture du programme.

La structure arborescente de IMALGO reproduit alors la structure du programme. Lorsque deux fonctions se succèdent, elles constituent des noeuds frères, lorsqu'une fonction en appelle une autre, elles sont reliées par un lien du type père-fils.

L'examen du programme nous fournit donc une structure représentant l'enchaînement et l'imbrication d'actions qui forment les noeuds d'un arbre tel que celui schématisé ci-dessous :

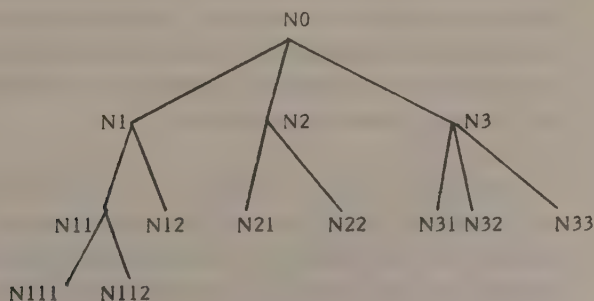


Fig II.4 Structure de IMALGO

Les différents nœuds N_i représentent successivement les différentes fonctions du programme. Par exemple, la fonction en N_1 peut appeler successivement les fonctions représentées par N_{11} et N_{12} , et est suivie de N_2 .

Dans IMALGO, sont représentées toutes les possibilités que le programme RAISON est capable de traiter. De ce fait, la succession de deux branches ne signifie pas toujours que celles-ci conduisent à des actions successives du programme. Elles peuvent aussi représenter deux fonctions qui sont les deux termes d'une alternative et, de ce fait, sont mutuellement exclusives. Nous verrons plus loin que cette ambiguïté disparaît naturellement dans les représentations IMEXEC.

Nous allons voir maintenant quelles sont les connaissances qui sont associées à cette représentation, à savoir ce que sont réellement les actions symbolisées par les nœuds de l'arbre et quelle est leur finalité.

II.3.2.2 - Les connaissances représentées au niveau de IMALGO

Nous voulons associer à chaque nœud de IMALGO la description de l'action réalisée par la fonction symbolisée par ce nœud. Nous avons choisi de faire appel au programmeur pour obtenir cette donnée. En fait, nous utilisons pour cela, les commentaires écrits pour décrire le programme. Le but de ces commentaires est d'expliquer ce que "fait" chaque fonction. Nous avons donc décidé de les utiliser pour constituer ce que nous avons appelé le "nom" des nœuds, à savoir la description de l'action correspondante.

Pour que notre système puisse fournir des explications cohérentes, il nous faut donc travailler sur des programmes commentés. C'est, bien sûr, une contrainte imposée au programmeur. Toutefois, pour en amoindrir les effets, nous envisageons de rendre cette procédure interactive. Nous pourrions alors demander au program-

meur d'expliquer le but d'une fonction quand celle-ci n'est pas précédée d'un commentaire, ou de valider la description lorsque le commentaire existe.

De plus, pour rendre cette représentation plus fiable, nous pouvons ensuite appliquer à IMALGO une procédure analogue aux procédures d'explication (cf §III.3) pour engendrer une paraphrase de l'algorithme. Si cette paraphrase ne le satisfait pas, le programmeur peut alors modifier ses commentaires et recommencer la construction de IMALGO, jusqu'à obtention d'un résultat correct.

Les commentaires vont donc nous permettre d'identifier les noms des nœuds représentant les différentes fonctions du programme.

Une remarque importante est à faire au sujet de ces noms. La plupart des actions d'un programme ont pour but d'examiner ou de modifier les valeurs des données ou des résultats intermédiaires obtenus. Dans les commentaires vont donc figurer certains noms de variables qui sont utilisées par le programme. La présence de ces variables nous permet de prendre en compte la partie déclarative du fonctionnement du programme : si on prend l'exemple des systèmes experts, nous pouvons ainsi intégrer dans IMALGO à la fois une représentation de leur fonctionnement procédural, en montrant comment sont choisies les règles à activer, et leur partie déclarative en intégrant la formulation de ces règles dans les noms des nœuds.

Voyons l'exemple d'un arbre IMALGO obtenu avec le programme RAISON utilisé dans notre application des pages jaunes.

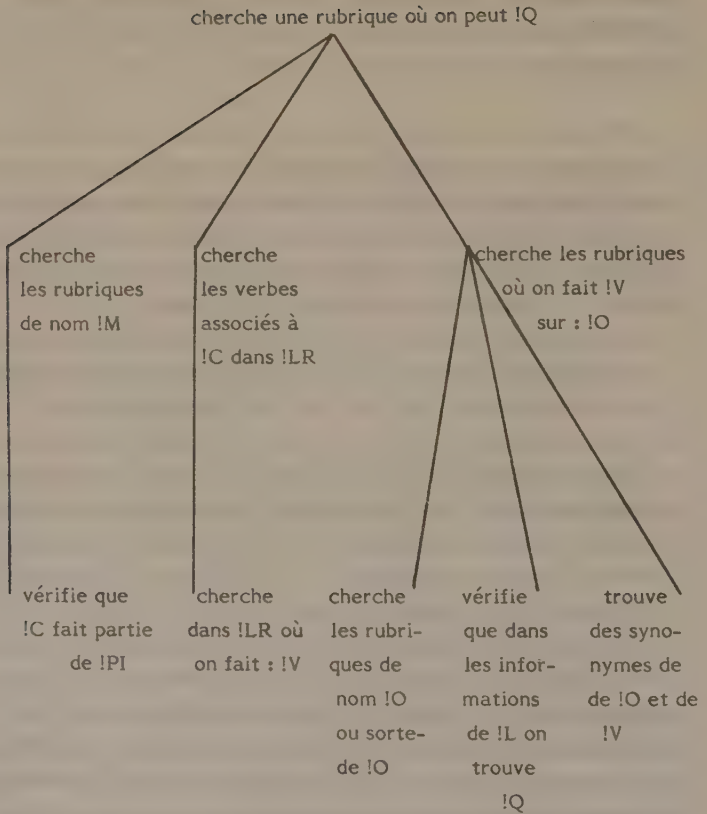


Fig II.5 L'arbre IMALGO obtenu avec l'application "pages jaunes"

Le schéma traduit que l'application "pages jaunes" cherche une rubrique où on peut !Q, c'est-à-dire le thème de la requête de l'utilisateur. Pour cela, on commence par chercher des rubriques dont le nom contient le métier !M. C'est le cas lorsque la requête !Q correspond à un professionnel !M.

Dans le cas où la demande n'est pas sur un métier, on recherche les verbes associés au concept !C dans la liste des rubriques !LR. C'est le cas lorsque la requête se réduit à un concept !C qui n'est pas un professionnel; pour cela on vérifie ensuite que le concept !C figure bien dans les informations pragmatiques !PI de chaque rubrique de !LR.

Un troisième type de recherche est envisagé lorsque la requête !Q consiste à rechercher où une action !V est effectuée sur un objet !O. C'est le troisième type de demande acceptée par RAISON. On voit alors que, pour réaliser cette action, nous en envisageons quatre, que nous ne détaillons pas ici.

On retrouve bien dans cet arbre, le principe de l'algorithme de RAISON.

Nous ferons une dernière remarque sur l'organisation de IMALGO.

Pour des raisons de commodité lors de l'utilisation ultérieure des noms des nœuds, nous les conservons sous forme de représentation interne analogue à celle décrite précédemment (§II.2).

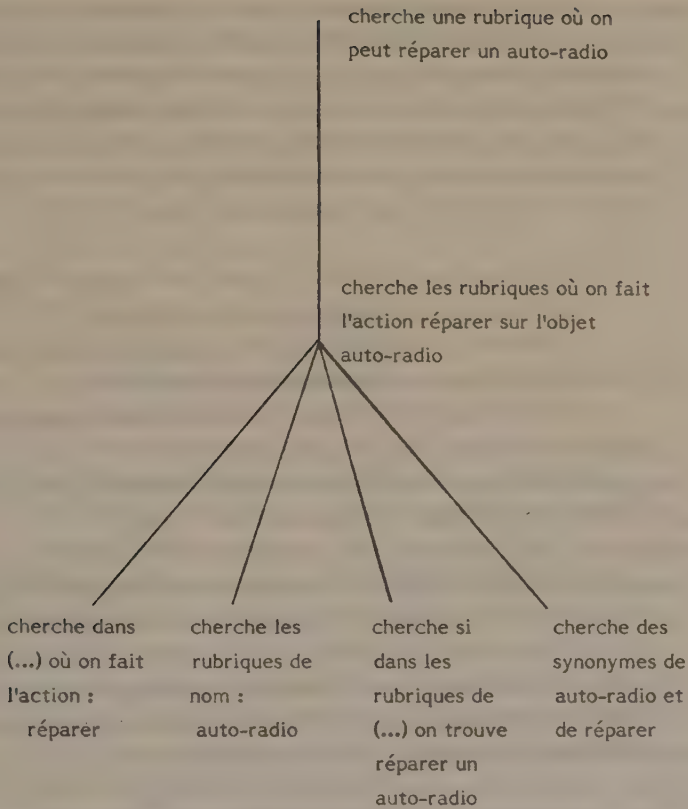
L'analyse a lieu lors de la création de IMALGO, afin de rendre plus rapide la création des arbres correspondants à une exécution (IMEXEC). La création de IMALGO ayant lieu avant l'utilisation réelle du système, le temps mis pour la réaliser est un critère beaucoup moins contraignant, puisqu'il n'influe pas sur la durée des interactions utilisateur-système. Un problème se pose lors de l'analyse avec les noms de variable du programme. Elles ne font pas partie du lexique reconnu par l'analyseur. Pour éviter un blocage, les variables figurent codées dans le commentaire afin d'être identifiées. (!Q, par exemple dans la Fig II.5). L'analyseur se contente alors de transférer ces éléments dans la structure interne sans les modifier. Nous reviendrons plus loin sur les procédures mises en place pour aboutir à ce résultat.

II.3.3 IMEXEC : la représentation d'une exécution du module RAISON

Lors de chaque exécution du module RAISON, nous construisons une trace du travail qu'il a effectué. RAISON est appelé pour effectuer une recherche dans la base de connaissances afin de répondre à une requête de l'utilisateur. Au cours de chaque exécution, une représentation IMEXEC est automatiquement créée. Celle-ci reprend une partie de la représentation IMALGO décrite précédemment. Cette représentation est alors précisée : les actions réalisées dans les différentes fonctions du programme sont complètement explicitées à l'intérieur des noms des nœuds où les variables sont remplacées par leurs valeurs lors de l'exécution.

On retrouve ainsi la démarche suivie par le programme RAISON pour obtenir les éléments d'une réponse en fonction d'une question particulière.

Voyons un exemple d'un arbre IMEXEC obtenu dans le cadre de l'application "pages jaunes". La requête de l'utilisateur étant : "Je veux réparer mon autoradio".



(...) : liste trop longue pour figurer ici en entier

Fig II.6 Exemple de IMEXEC

Dans cette représentation, on retrouve donc l'aspect hiérarchique de la structure IMALGO, avec l'évaluation exacte des noms des actions réalisées par le module RAISON.

Soulignons l'intérêt de cette structure par rapport à une trace classique : la plupart des systèmes d'exploitation présents dans les différents types de machine, sont en mesure de fournir une "trace" de l'exécution d'un programme; mais, dans celle-ci, on ne peut trouver que la liste des différentes fonctions, définies par le programmeur, qui ont été invoquées durant l'exécution, avec la valeur de leurs paramètres propres lors de cet appel.

Cette liste est donc difficile à exploiter pour retrouver exactement ce que le programme a fait :

- une fonction peut travailler à la fois sur des paramètres qui lui sont spécifiques et sur des données plus générales;
- le nom d'une fonction est loin d'être suffisant pour savoir quelle en est l'action précise;
- la limitation aux seules fonctions définies dans le programme peut être trop restrictive.

Dans la structure IMEXEC, on retrouve une structure arborescente analogue à celle que peut fournir une fonction de traçage, mais les nœuds ne sont pas obligatoirement identiques : certains nœuds supplémentaires peuvent exister, par contre d'autres nœuds peuvent ne pas avoir été pris en compte lors de la construction de IMALGO (absence de commentaires, fonctions très simples...)

L'avantage le plus important de cette représentation se situe dans les noms des nœuds. Ils présentent de façon explicite le travail exécuté par la partie du programme concernée en fonction de la valeur des données qui lui sont fournies.

Dans un arbre IMEXEC, on peut retrouver plusieurs nœuds successifs qui sont l'image d'un même nœud de IMALGO, mais avec des noms différents. Cette répétition est due au caractère itératif de certains passages du programme.

Les variantes dans les noms des nœuds indiquent les changements de valeur des variables lors de chaque passage. Nous détaillerons dans le chapitre III comment nous avons traité la représentation des itérations, lors de la construction de IMALGO, (en particulier l'utilisation de récursivité fréquente dans un programme Lisp), et comment nous avons différencié les nœuds correspondants à une même itération dans IMEXEC.

II.4 - La représentation du thème

Définissons tout d'abord le sens que nous donnons au mot thème.

Comme nous l'avons dit plus haut (voir §1.3.1), nous ne reprenons pas ici la définition donnée en Linguistique de phrase, où la notion de thème vient en opposition avec le rhème. Dans notre travail sur le dialogue, nous avons adopté une définition beaucoup plus large. Pour nous, le thème est le sujet du dialogue, ce dont on est en train de parler.

Pour cette partie de notre travail, nous avons utilisé les résultats obtenus par B.Grau (Grau 83) sur l'analyse et la représentation d'un texte. Son système repose sur la notion de thème du discours. Par le suivi de l'évolution du thème, la cohérence du texte est vérifiée, et chaque phrase est intégrée dans son contexte. Un thème est représenté par un ensemble de schémas qui le décrivent : ces schémas sont reliés entre eux par des liens de différents types (hiérarchiques, descriptifs,...).

Nous ne reviendrons pas ici sur la définition et l'organisation détaillée des schémas définis dans le système de B.Grau; nous nous contenterons d'exploiter les résultats que son système permet d'obtenir. Nous nous intéressons plus particulièrement aux aspects liés à l'intégration d'une phrase dans son contexte, c'est à dire au rattachement d'une phrase aux thèmes précédemment abordés.

Reprenant une représentation analogue à celle de ce système, nous représentons un thème par un couple (prédicat, objet) qui constitue le nom du schéma correspondant à ce thème. L'objet permet alors de préciser la portée du prédicat. Nous en avons montré des exemples dans §1.3.1.

Au cours du dialogue, les différents thèmes sont développés à la fois par l'utilisateur lorsqu'il introduit sa requête ou qu'il l'élargit, et par le système lorsque le module RAISON effectue des approximations. Parmi tous ces thèmes, il existe une hiérarchie : elle traduit le fait qu'un certain nombre de déviations dans les différents thèmes développés peuvent se produire à la suite d'une requête de l'utilisateur, sans pour autant que l'on change complètement de sujet.

Cette hiérarchie sera traduite par la création d'un arbre, dont la racine est la requête initiale de l'utilisateur.

Nous considérons que le module RAISON ne peut pas introduire un thème entièrement nouveau dans le type d'application que nous traitons, mais il peut le dévier. Le premier type de déviation que le module RAISON peut introduire, est celui qui consiste à passer du thème présenté dans la question de l'utilisateur à celui qui constitue la réponse trouvée quand ils ne coïncident pas rigoureusement.

Par exemple, dans l'application "pages jaunes", si la requête de l'utilisateur est : "Je voudrais faire réparer ma voiture", le système trouve la rubrique garagiste, et la propose à l'utilisateur. On passe donc du thème : "réparer voiture" à celui de "être garagiste".

Remarquons au passage que, lorsqu'il est question d'un professionnel sans autre précision, nous considérons qu'il s'agit du thème "être professionnel".

D'autres déviations sont envisageables lorsque le système doit effectuer des approximations.

Le couple (prédicat, objet) est alors transformé. Un seul terme peut être modifié : il s'agit alors de l'application d'une synonymie approchée sur l'objet ou sur le verbe (voir (Sabah 84)). Dans d'autres cas, le couple est entièrement modifié, le module RAISON a dû effectuer une transformation admissible sur chacun des deux termes pour trouver un résultat dans la base de connaissances. Dans l'un ou l'autre cas, le changement de thème n'est pas complet, il ne s'agit que de déviations à partir d'un même thème qui figurait dans la requête de l'utilisateur.

L'utilisateur peut lui aussi introduire une nouvelle déviation en questionnant le système, sur une réponse qui lui a été fournie, donc sans changer de sujet. Nous utilisons alors les procédures mises en place par B.Grau pour rattacher la question de l'utilisateur à l'un des thèmes précédemment développés. Si le rattachement est possible, on indique alors un lien entre ces deux thèmes. Par contre, si le thème d'une question ne peut être rapproché d'aucun autre déjà traité, il s'agit alors d'une requête concernant un sujet nouveau. Elle nécessite un appel au module RAISON pour la recherche d'une réponse dans la base de connaissances. Ce changement de sujet provoque la création d'une nouvelle arborescence dont la racine est cette dernière requête.

Les différents thèmes traités dans un dialogue vont donc constituer une liste d'arborescences. Prenons un exemple.

L'utilisateur pose comme première requête :

"Je voudrais réparer mon auto-radio"

Un arbre est alors créé dont la racine est "réparer auto-radio", et RAISON est appelé avec ce même couple.

Les approximations faites par RAISON pour trouver une réponse satisfaisante vont alors provoquer l'ajout de nouveaux thèmes dépendant de (réparer auto-radio). Ces thèmes sont : (réparer appareil

électrique), (réparer automobile), (vendre auto-radio),... On retrouve ces différents thèmes en examinant les noms des nœuds de IMEXEC correspondant à la recherche effectuée par RAISON.

On obtient alors l'arbre des thèmes suivant :

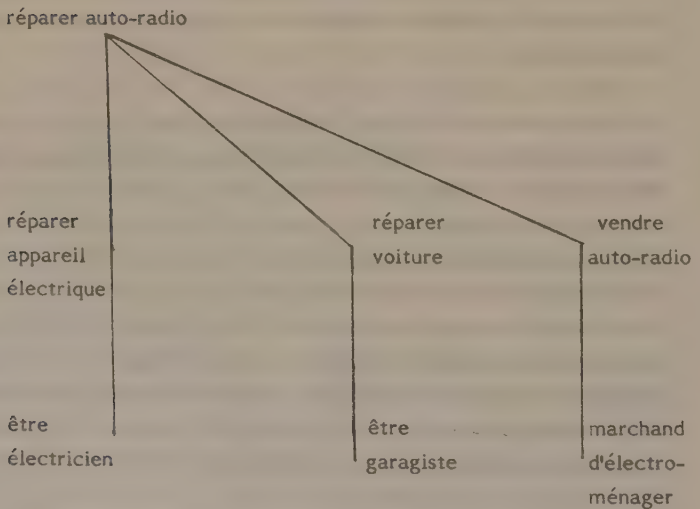


Fig II.7 - Un exemple d'arbre de thèmes

Si dans la suite du dialogue l'utilisateur demande pourquoi on lui a proposé un mécanicien, on pourra rattacher le thème (être mécanicien) à (être garagiste). Ayant retrouvé le contexte dans lequel la demande de l'utilisateur prend son sens, on pourra alors lui répondre correctement.

Un problème particulier se pose lorsque, dans la requête de l'utilisateur, on ne trouve pas un thème complètement défini : l'un des éléments du couple (prédicat, objet) n'est pas fourni. On essaie alors

de le rattacher à un autre thème et on infère qu'il est toujours question du même sujet. Lorsque cette procédure échoue, le module RAISON est activé avec cette demande incomplète. Nous verrons plus loin comment RAISON tente de trouver une solution ou une liste de solutions possibles lorsqu'il est placé dans de telles circonstances.

II.5 - La représentation de l'utilisateur

Pour intervenir correctement au cours de l'échange, il nous faut savoir qui sont les utilisateurs, quelles sont leurs connaissances, et quel est le problème particulier de chaque utilisateur.

Pour cela, nous avons distingué deux sortes de représentation. D'une part, nous avons essayé d'établir quel était le niveau moyen des connaissances d'un utilisateur potentiel sur le domaine de l'application choisi. Le but est de répondre à ses questions en restant suffisamment clair sans être trop détaillé. Cette représentation est activée pour chaque utilisateur au début du dialogue, et pourra être modifiée au cours de l'échange.

Par ailleurs, nous avons essayé de mettre en place une représentation du problème particulier d'un utilisateur donné, afin de répondre plus spécifiquement à sa demande. Cette représentation est alors entièrement construite au cours du dialogue; elle est déduite directement des interventions de l'utilisateur.

II.5.1 - Les connaissances présumées sur le domaine d'application

Cette représentation nous est utile pour savoir "que dire" à l'utilisateur quand il demande des explications sur ce que le système lui a dit. De ce fait, elle est liée à la représentation des actions du système. Elle est alors à rapprocher de la représentation IMALGO. Pour savoir quelles action il est inutile d'expliquer à l'utilisateur, il nous faut associer à chaque action du système, c'est à dire à chaque

noeud d'IMALGO, une notion de "niveau de généralité". Par ces termes, nous voulons signifier une certaine mesure de l'importance de l'action exécutée par le système. Cette mesure est liée à la structure hiérarchique de IMALGO. Nous voulons en fait localiser à quel niveau, dans chacune des branches de l'arbre IMALGO, se trouve une action, pour la compréhension de laquelle, l'utilisateur n'a pas besoin de plus de détails.

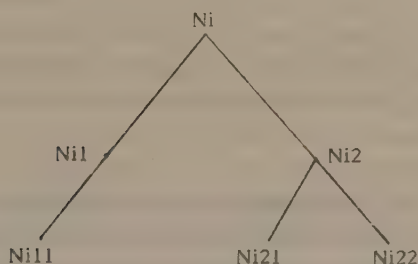
On considère alors que les nœuds qui se trouvent sous le nœud considéré sont implicites pour un utilisateur potentiel. On présuppose ainsi chez l'utilisateur les connaissances nécessaires pour faire lui-même ces déductions.

Notre représentation consiste à associer à un utilisateur une liste de pointeurs vers des nœuds de IMALGO. Chaque pointeur désigne une action qui peut être considérée comme terminale : il est alors inutile de développer les actions (c'est à dire les nœuds) qui en dépendent afin d'expliquer son rôle dans le raisonnement. Les actions qui ne sont pas prises en compte sont celles que l'utilisateur peut déduire.

Cette représentation est assez simpliste pour l'instant, mais elle n'est pas figée. En effet, si l'utilisateur ne parvient pas à faire les déductions voulues, la représentation de ses connaissances, qui était donnée a priori, est modifiée afin de prendre en compte son incompréhension. La modification consiste alors à déplacer les pointeurs vers des nœuds plus profonds de IMALGO.

Cette évolution se traduit de deux façons, suivant le besoin d'informations supplémentaires manifesté par l'utilisateur. Quand la demande d'explication porte sur un point particulier, on descend d'un certain nombre de niveaux dans une branche de l'arbre, et la liste des pointeurs n'est modifiée que partiellement : dans cette branche, le nœud au dessous duquel on ne descendait pas est remplacé par une liste de ses descendants.

Par exemple, dans l'arbre suivant, si la question portait sur la i^{ème} branche, dans laquelle on s'arrêtait au nœud Ni, on descendra ensuite d'un niveau et ce nœud sera remplacé dans la liste des pointeurs par ses deux fils Ni1 et Ni2.



Par contre, si le besoin d'information supplémentaire est général, on appliquera une procédure analogue à l'ensemble des branches.

La procédure décrite ici peut se répéter tant qu'on est pas arrivé aux feuilles de IMEXEC.

Nous montrerons dans le chapitre III comment nous utilisons pratiquement ces principes.

La représentation des connaissances d'un utilisateur particulier peut ainsi être précisée au cours de l'échange : le système dispose au départ d'un certain "modèle" de l'utilisateur potentiel, et, en fonction des questions posées, cette représentation évolue pour prendre en compte le niveau des connaissances propres à chaque utilisateur. Ceci correspond à une notion de "belief" embryonnaire sur l'utilisateur de la part du système (Bien 83, Steel 84).

Dès qu'un arbre IMEXEC est créé, quand une recherche a donc été menée dans la base de connaissances, les pointeurs vers IMALGO qui indiquaient les connaissances a priori d'un utilisateur sont transférées vers IMEXEC : au lieu de pointer vers IMALGO, ils pointeront vers les nœuds de IMEXEC, images des nœuds de IMALGO.

Nous pensons qu'une étude plus approfondie des caractéristiques des utilisateurs d'un système donné, devrait permettre d'améliorer cette représentation. On pourrait alors mieux définir quel est le niveau moyen des connaissances de l'utilisateur habituel sur le sujet concerné.

Pour l'application que nous avons réalisée, nous avons adopté une méthode tout à fait empirique pour apprécier le niveau moyen des connaissances d'un utilisateur. Il faudrait tester ce système auprès d'un public d'utilisateurs afin de modifier et d'affiner cette représentation.

On pourrait également mieux utiliser ce que dit l'utilisateur et comment il formule ses requêtes pour inférer son niveau de connaissances sur un sujet. Les problèmes de "visée vers l'auditoire" (cf Clark 82) pourrait alors être mieux pris en compte.

II.5.2 - Comment représenter le problème particulier d'un utilisateur

Nous voulons ici essayer de prendre en compte l'ensemble du problème posé par l'utilisateur lorsqu'il expose sa requête. Nous rejoignons ici les préoccupations de (Allen 80) et (Perrault 78), en voulant essayer de trouver précisément le but de l'utilisateur.

Le fait de pouvoir déduire qu'un utilisateur qui demande l'heure d'un train, souhaite également savoir son quai de départ, nécessite une définition précise du domaine d'application. Mais nous avons considéré que l'utilisateur posait toujours une question ou soumettait un problème (voir §III.2.1).

Toutefois, nous restons en retrait par rapport aux travaux d'Allen et Perrault, dans la mesure où nous avons voulu mettre en place des procédures qui ne soient pas spécifiques à un système de questions-réponses particulier. Nous n'avons donc pas pu utiliser, comme l'ont fait ces auteurs, des connaissances concernant le domaine d'application pour analyser les intentions de l'utilisateur.

Nous avons vu que, dans l'appel à RAISON, DIALOG ne considère que le thème de la requête exposée. Nous voulons ici représenter la partie "oubliée" de la question de l'utilisateur. Ce sont donc les inflexions qui figurent dans la requête. Ces inflexions sont les éléments qui existent dans la représentation interne de la question, mais qui ne forment pas le couple (prédicat, objet) utilisé pour activer RAISON. Dans notre représentation fondée sur la grammaire de cas de Fillmore, ce sont les cas qui ne sont pas l'objet du prédicat. Il peut s'agir de concepts simples ou de phrases complètes venant remplir l'un de ces cas (voir Fig II.1).

Cette représentation nous permet, au cours du dialogue, de résoudre certains problèmes d'ambiguïtés. Quand la recherche menée par RAISON avec seulement le thème de la question, ne permet pas de trouver une réponse acceptable, nous utiliserons ces données supplémentaires pour lever l'ambiguïté, avant de poser une question à l'utilisateur. Nous verrons dans le chapitre suivant quelles sont les procédures mises en place pour aboutir à ce résultat.

II.6 -L'historique du dialogue

1

L'historique du dialogue représente le lien entre les différentes représentations que nous venons d'évoquer.

C'est grâce à lui que nous savons ce qui s'est passé au cours du dialogue. Il assure le suivi des différentes interventions, et permet ainsi de savoir "qui" a dit "quoi", et dans quel contexte une intervention a été prononcée.

Il va se présenter sous forme d'un répertoire dans lequel chaque "page" correspond à une intervention au cours de l'échange. La structure de chaque page va être identifiée par le numéro de cette intervention.

Voici un schéma du contenu d'une page de l'historique.

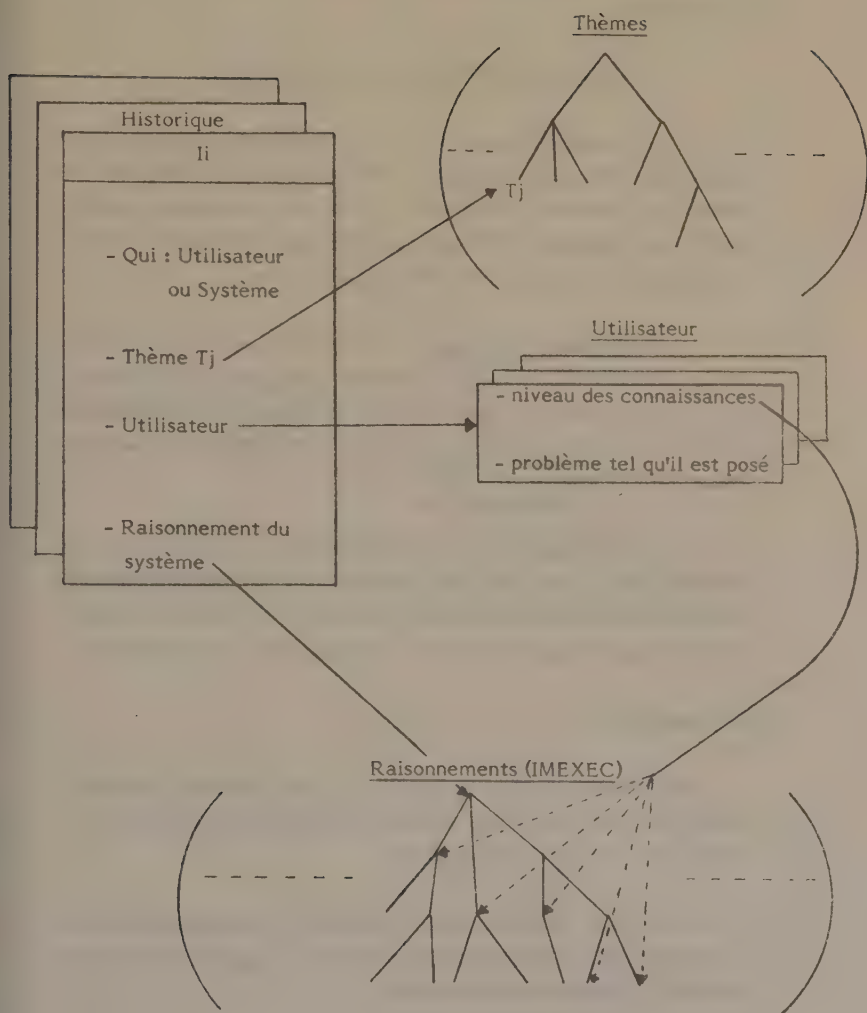


Fig II.8 L'historique du dialogue

Dans cette représentation, on trouve donc, à la suite de l'intervention :

- qui l'a prononcée (l'utilisateur ou le système).

- le thème développé dans cette intervention : il s'agit alors d'un pointeur vers un élément de la liste des arbres représentant les thèmes. Cet élément peut être la racine d'un des arbres, quand il s'agit d'une requête de l'utilisateur sur un sujet nouveau, ou un nœud quelconque de l'un de ces arbres dans les autres cas.

- l'utilisateur : il s'agit ici d'un pointeur vers la représentation de l'utilisateur, tel qu'il se présente à cet instant de l'échange; donc à la fois vers une représentation du niveau des connaissances qu'on lui suppose à cet instant, et vers la représentation du problème qu'il se pose tel qu'on a pu le définir.

- le raisonnement fait par le système : il s'agit alors d'un pointeur vers un arbre IMEXEC construit lors du fonctionnement de RAISON. Ce pointeur n'est présent que dans le cas d'une intervention du système pour laquelle ce dernier a fait appel au module RAISON.

II.7 - Conclusion

Nous avons présenté dans ce chapitre l'ensemble des représentations qui nous permettent de suivre à la fois l'évolution du dialogue et les attitudes des deux interlocuteurs.

Nous allons voir maintenant comment ces représentations sont constituées puis utilisées afin de réaliser un dialogue respectant les principes présentés dans le chapitre I.

- CHAPITRE III -

Les procédures

Dans ce chapitre nous allons voir comment les différentes représentations que nous avons introduites dans le chapitre précédent sont mises en place et complétées au cours du dialogue.

Nous verrons d'abord la création de IMALGO, qui a lieu avant la mise en œuvre du système, puis la création et la mise à jour des autres représentations.

Nous présenterons ensuite les procédures d'explication et de relance du dialogue qui utilisent ces représentations, afin de permettre au système de respecter les principes présentés dans le premier chapitre.

III.1 CONSTIM : création de IMALGO

Nous avons vu dans le chapitre précédent (§ II.3) que la création de IMALGO se fait par l'application d'une procédure préalable au programme RAISON.

On peut schématiser ainsi cette procédure :

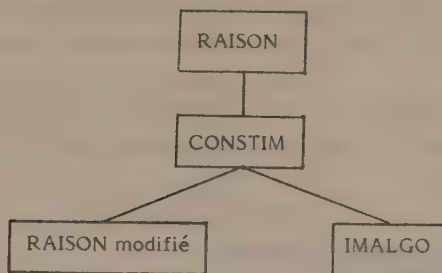


Fig. III.1 : la procédure CONSTIM

Le but de cette procédure est d'extraire la structure du programme RAISON, afin de construire un arbre IMALGO la représentant, et d'ajouter à RAISON les procédures permettant de construire les arbres IMEXEC. CONSTIM prend comme donnée le programme RAISON et donne en résultats le programme modifié et l'arbre IMALGO.

III.1.1) Le principe adopté

La structure arborescente est construite de la façon suivante : le nœud racine symbolise le programme complet, chaque nœud représente une fonction.

Quand une fonction f appelle une fonction g , g constitue un nœud fils de f : on augmente ainsi la profondeur de l'arbre.

Quand deux appels sont consécutifs, c'est à dire quand ils constituent deux instructions qui se suivent, ces deux fonctions sont considérées comme des frères et sont situées à un même niveau dans l'arbre.

Pour écrire cette procédure, nous avons pris en compte les caractéristiques du langage de programmation utilisé dans RAISON, en l'occurrence VLisp pour l'application sur laquelle nous avons travaillé. Nous nous sommes inspiré des principes utilisés par Wertz pour la correction de programme de débutants (Wertz 78).

RAISON, quant à lui, est modifié, en ce sens qu'on lui a adjoint la possibilité de construire des arbres IMEXEC, images partielles d'IMALGO.

III.1.2) La réalisation pratique

La première étape de ce travail consiste à définir la liste des différentes fonctions définies dans le programme RAISON.

Ensuite, on parcourt le programme en constituant un niveau supplémentaire, dans une branche de l'arbre, à chaque appel d'une fonction.

Le noeud "racine" est celui qui symbolise le programme complet. On crée IMALGO en profondeur d'abord, et de gauche à droite.

Dans l'application sur les pages jaunes de l'annuaire, nous avons travaillé en Lisp, et un appel se traduit alors par la rencontre du nom d'une fonction dans une liste.

Par exemple, si on a défini une fonction RECHRUB dans laquelle on rencontre : (RUBNOM M) et que RUBNOM est également une fonction définie, on créera, pour représenter RUBNOM, un nœud fils du nœud symbolisant RECHRUB.

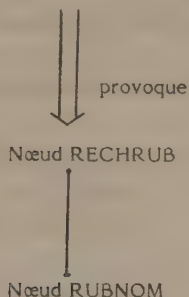
(DE RECHRUB (...) (cherche une rubrique)

(...)

(RUBNOM M)

(cherche une rubrique contenant M dans son nom)

(...))



Quand on rencontre ainsi un appel à une fonction, on suspend l'examen de la fonction "père" pour passer au parcours de la fonction "fils". Si on reprend l'exemple précédent, on interrompt la lecture de la fonction RECHRUB pour parcourir RUBNOM où on applique la même technique.

On peut également rencontrer un appel de fonction dans les paramètres d'une fonction prédéfinie ou non. Ainsi on rencontre des formes telles que :

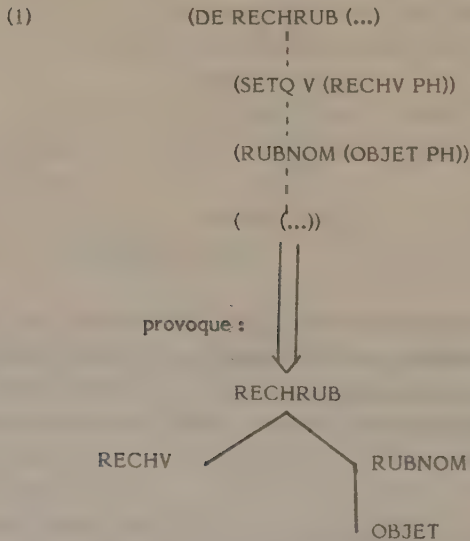
(SETQ RUB (RUBNOM M))

ou (RUBNOM(OBJET PH)) (cherche une rubrique dont le
nom est l'objet de la phrase PH)

Dans le cas où il s'agit d'une fonction prédéfinie, on considère la fonction rencontrée comme un fils (de façon analogue au cas précédent).

Par contre, quand une fonction figure comme paramètre d'une autre fonction, on la considère alors comme un fils de cette dernière fonction (et donc un "petit-fils" de la fonction dans laquelle on l'a rencontrée).

Ainsi un extrait de code tel que celui-ci :



On pourrait contester notre méthode; en effet, il est possible d'écrire la dernière fonction de la manière suivante :

(2) (SETQ O (OBJET PH))
 (RUBNOM O)

et on obtiendrait :



On aurait donc pu construire ce type d'arbre également à partir de la forme (1). Mais nous pensons que notre représentation est préférable : si l'appel à la fonction OBJET figure comme paramètre dans l'appel de RUBNOM, c'est vraisemblablement parce que cette fonction n'est utile que pour le travail effectué par RUBNOM. La formulation (2) est préférée quand on doit utiliser les résultats obtenus par cette fonction au niveau de RECHRUB, et non pas seulement dans RUBNOM.

Un problème particulier se pose pour l'usage de la récursivité. Nous avons alors choisi de ne constituer qu'un nœud pour représenter cette fonction.

```
Ainsi : (DE RUBNOM (M,L)
          |
          (RUBNOM M (CDR L)))
```

ne provoque pas la création d'un nœud fils de RUBNOM, qui représenterait à nouveau RUBNOM.

Par contre, nous avons considéré que l'appel d'une même fonction à des endroits différents du programme, provoque la création de deux nœuds distincts. On transfère alors également les descendants de ce nœud.

A la création d'un nouveau nœud de IMALGO, on ajoute à la description de ce nœud une identification. Celle-ci est constituée à partir du commentaire situé au début de la fonction, comme nous l'avons vu dans le chapitre précédent.

Pour cela, on recherche ce commentaire juste après l'identification de la fonction (après l'instruction (DE nom de fonction) ou (DF nom de fonction) en Lisp). Quand celui-ci n'existe pas, on continue le parcours de la fonction, pour rechercher d'autres nœuds éventuels. Quand on les rencontre, on se contente alors de les attacher au nœud "père" de celui que l'on aurait dû créer, si on avait pu lui trouver une identification.

C'est ici que nous proposons d'apporter une éventuelle amélioration en rendant la procédure CONSTIM interactive, ceci nous permettrait alors de vérifier qu'il s'agit bien d'un nœud dont le programmeur estime qu'il est inutile de conserver la trace dans IMALGO. L'absence de commentaire provoque donc l'annulation d'un nœud dans l'arbre IMALGO.

Ajouter une identification au nœud, consiste alors à extraire le commentaire décrivant l'action de la fonction, à l'analyser tout en conservant les variables codées qui figurent dans ces commentaires, et à l'ajouter comme nom dans la p-liste de l'atome que constitue le nœud correspondant.

Au cours de la construction d'IMALGO, on modifie le programme RAISON afin de lui fournir les moyens de construire les arbres IMEXEC par la suite.

Cette modification consiste à insérer au début de chaque fonction des instructions. Le but de ces instructions est de provoquer la construction d'un nouveau nœud dans l'arbre IMEXEC, image de celui qui vient d'être constitué dans IMALGO. Elles permettent également de provoquer l'évaluation des variables qui figurent dans l'identification des nœuds.

III.2 Le suivi des représentations

Nous allons présenter ici les procédures mises en place afin de gérer les représentations du thème, de l'utilisateur et de l'historique que nous avons décrites dans le chapitre précédent.

III.2.1 La création des arbres représentant le thème

Un thème peut être introduit soit par l'utilisateur, soit par le système quand il fait un raisonnement.

Examinons tout d'abord le cas où c'est l'utilisateur qui introduit un thème par son intervention.

1) Le thème introduit par l'utilisateur

Chaque intervention de l'utilisateur est analysée et son thème est mis en évidence. Pour cela, nous utilisons les travaux réalisés par M.Rady (Rady 83) et B.Grau (Grau 83). L'analyse produit alors la représentation interne de la phrase quand cela est possible. Nous ne prendrons pas ici en compte les phrases sur lesquelles l'analyseur échoue. Les améliorations à apporter à l'analyseur afin de prendre en compte ces phénomènes font l'objet du travail de J.B.Berthelin (Berthelin 83), et nous ne les évoquerons pas ici.

A partir de cette représentation, nous pouvons alors déterminer le thème de la phrase, en cherchant le schéma correspondant dans nos connaissances. Suivant le type de l'intervention, nous appliquons alors des procédures différentes. La motivation (cf §II.1) indique qu'il s'agit d'une information ou d'une demande.

A) L'intervention est une information

Dans notre travail, nous nous sommes intéressé au problème du dialogue dans un système de questions-réponses. Notre but n'est donc pas d'essayer d'enrichir la base de connaissances à partir d'informations fournies par l'utilisateur au cours du dialogue. Ceci constituerait un autre sujet de recherche, sur l'acquisition automatique de connaissances à partir du dialogue.

Le plan que nous supposons a priori chez l'utilisateur consiste à obtenir des réponses, ou des solutions à ses problèmes. Nous n'essaierons donc pas d'interpréter ces informations comme de nouvelles données, nous ne les prenons en compte que dans la mesure où elles présentent un problème auquel RAISON est en mesure de trouver une solution. Nous la transformons alors en une demande concernant un professionnel capable de résoudre le problème posé.

Un autre type d'information est pris en compte : quand il s'agit d'une réponse formulée à la suite d'une demande du système. Nous devons alors être en mesure de l'attacher à l'un des thèmes précédemment abordés.

Rappelons que nous parlons ici d'un professionnel en nous situant dans l'application sur laquelle nous avons travaillé, mais nous pensons que cette stratégie est adaptable aux autres systèmes de questions-réponses. En restant à l'intérieur des limites que nous nous sommes fixées, nous ramenons donc le cas des informations à celui des demandes "objets" concernant un professionnel, les autres types d'informations n'étant pas traitées ici.

B) L'intervention est une demande

La motivation (au sens défini dans (Sabah 76) et (Rady 83)) permet de déterminer que la requête est une demande, et signale qu'il s'agit : d'une demande objet (du type "qui...", "que..."), d'une demande causale (du type "pourquoi...?") ou d'une demande polaire (appelant une réponse par oui ou par non).

Nous n'aborderons pas dans ce travail, le cas de la demande polaire. En effet, comme nous venons de le voir, dans notre application, le but est de trouver un professionnel susceptible de résoudre un problème posé par l'utilisateur. RAISON ne peut donc pas, a priori, répondre à une question par oui ou par non. De ce fait, DIALOG ne l'active pas dans le cas d'une demande polaire.

Il est possible que dans d'autres applications (de renseignement, par exemple), RAISON soit alors mis en œuvre. Le cas des demandes polaires serait alors traité comme celui des demandes objets. Nous aurions pu éventuellement transformer ce type de demande en demandes objets. Par exemple : "Est-ce qu'un garagiste répare des auto-radios ?" pourrait être transformé en une demande objet, de thème "réparer auto-radio", et DIALOG devrait alors comparer les

résultats avec la demande de l'utilisateur. Mais nous avons pensé que ce genre de transformation pouvait se révéler souvent plus complexe, et nécessiterait, pour être justifié, des études plus poussées. Nous avons donc choisi de ne pas traiter ce type de demande, dans le cadre de cette étude.

Nous n'examinerons ici que les deux premières hypothèses.

a) Demande objet

L'utilisateur initialise le dialogue avec ce type de demande.

Le thème de cette demande est alors le couple prédicat et objet sur lequel porte la demande. Nous essayons de l'attacher à un thème précédemment traité. En effet, une demande objet est, en principe, un appel à une recherche dans la base de connaissances sur un objet nouveau. En général cette procédure renvoie un résultat négatif.

Toutefois, quand cette recherche aboutit (il ne s'agit donc pas d'une requête sur un nouveau sujet), on retrouve le contexte dans lequel ce thème a déjà été abordé, et en particulier les recherches menées et les résultats que nous avons obtenus alors.

Nous complétons l'arbre développé en attachant ce thème à celui qui a été trouvé, c'est à dire que l'on ajoute une nouvelle branche aboutissant à un nœud représentant le thème de la requête de l'utilisateur.

On considère, en fait, que l'utilisateur n'a pas compris les résultats qui lui ont été transmis lors de la précédente intervention, ou n'en a pas admis la validité. Nous traitons donc sa requête comme une demande d'explications.

Quand la procédure de recherche ne reconnaît aucun des thèmes déjà abordés, on considère qu'il est question d'un nouveau sujet.

Nous créons un arbre dont la racine représente ce thème. Cet arbre, réduit pour l'instant à sa seule racine, est lui-même inclus dans la liste des différents sujets traités au cours du dialogue avec cet utilisateur. C'est le cas qui se produit, en particulier, à l'initialisation du dialogue.

b) La demande causale

Quand la motivation reconnue par l'analyseur est une demande causale, on commence également par chercher à attacher ce thème à un thème existant.

Une première remarque est à faire ici. Dans notre système, une demande causale ne peut être l'amorce d'un nouveau sujet. De ce fait, si la procédure n'aboutit pas, on ne peut répondre à ce type de demande.

Le programme RAISON ne peut répondre qu'à des requêtes portant sur des actions effectuées sur des objets, ou plus précisément, sur des personnes capables de réaliser ces actions.

Dans le cadre de notre application, nous n'avons considéré que des demandes sur des professionnels. Nous ne prenons pas en compte des questions telles que : "Pourquoi un garagiste répare-t-il des voitures ?", lorsqu'il n'a pas encore été question de ce sujet au cours du dialogue. De ce fait, nous pensons qu'il n'était pas justifié de transformer empiriquement ce type de demande (c'est-à-dire une demande causale) en demande objet. Il est, par contre, possible que ce type de modification soit admissible dans le cadre d'autres applications (où les demandes peuvent être des interrogations causales portant sur la base de connaissances). Il suffirait alors de modifier le module concerné dans notre programme.

Quand on a pu attacher le thème de la demande à un sujet déjà traité, on retrouve alors le contexte dans lequel ce thème a déjà été abordé. Ce peut être le même thème, ou un thème suffisamment voisin pour que l'attachement soit possible.

Reprenons l'exemple évoqué au (§II.4)

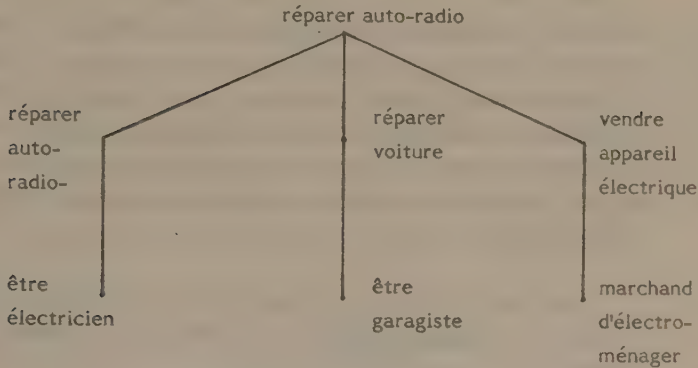


Fig II.7 un exemple d'arbre de thèmes

Cet arbre a été constitué à la suite de la requête : "Je voudrais réparer mon auto-radio".

Quand l'utilisateur demande, par exemple,

"Pourquoi me proposez-vous un mécanicien ?"

on peut rattacher le thème "être mécanicien" à "être garagiste". On retrouve alors tout le contexte correspondant à ce thème, c'est à dire l'arbre des différents thèmes qui ont été développés à partir du sujet : "réparer un auto-radio".

En parcourant l'historique, on retrouve également l'arbre IMEXEC qui a été créé à ce sujet. On peut ainsi chercher dans cet arbre ce qui a conduit à introduire le thème "être garagiste". Nous verrons plus loin comment fonctionne alors la procédure d'explication. Le but de la représentation des différents thèmes est atteint : nous avons retrouvé le contexte relié à chaque intervention.

2) Les thèmes introduits par le système

Nous avons vu que le système peut également provoquer des déviations de thème, tout en restant dans un même sujet. Ceci se produit lors de la recherche dans la base de connaissances. En effet, le but du programme RAISON est de trouver une réponse à la question de l'utilisateur, donc de passer du thème exprimé dans la requête, au thème qui en constitue la réponse. Ce phénomène est général, quel que soit le programme de recherche dans une base de connaissances que l'on considère.

Il est assez important dans l'application sur laquelle nous avons travaillé, en raison de l'utilisation des synonymies présentées dans (Sabah 84).

Les thèmes abordés par RAISON au cours de sa recherche peuvent alors être beaucoup plus nombreux. Pour les déterminer, nous allons examiner les arbres IMEXEC construits par RAISON. En particulier, nous allons étudier les noms des noeuds d'IMEXEC (c'est-à-dire l'interprétation des commentaires écrits dans le programme RAISON).

Nous allons parcourir les noeuds d'IMEXEC, et chercher les thèmes développés dans chacun de leur nom. Au fur et à mesure, chacun de ces thèmes viendra constituer un nouveau noeud dans l'arbre des thèmes dont la racine symbolise la requête de l'utilisateur.

Reprenons l'exemple de l'arbre IMEXEC (Fig II.6) obtenu à la suite de la requête : "Je voudrais réparer mon auto-radio".

A la réception de cette demande, nous avons constitué un nouvel arbre, inclus dans la liste des sujets traités. Cet arbre est alors réduit à sa seule racine qui est : "réparer auto-radio". Ce thème est utilisé par DIALOG pour faire appel à RAISON. Ce dernier, au cours de son

déroulement, construit l'arbre IMEXEC qui figure en Fig II.6 et le transmet à EXPLIQ, avant de rendre la main à DIALOG en lui envoyant les résultats qu'il a obtenus.

EXPLIQ entreprend alors la construction de l'arbre des thèmes correspondants. Le premier noeud a également pour thème "réparer auto-radio", EXPLIQ passe alors au noeud suivant. Au cours de cette construction la hiérarchie existant dans IMEXEC est reproduite. Ainsi le noeud de IMEXEC de nom "cherche où on fait réparer", dont le thème se réduit à "réparer" provoque la création d'un noeud fils de "réparer auto-radio" dans l'arbre des thèmes. On aura alors :

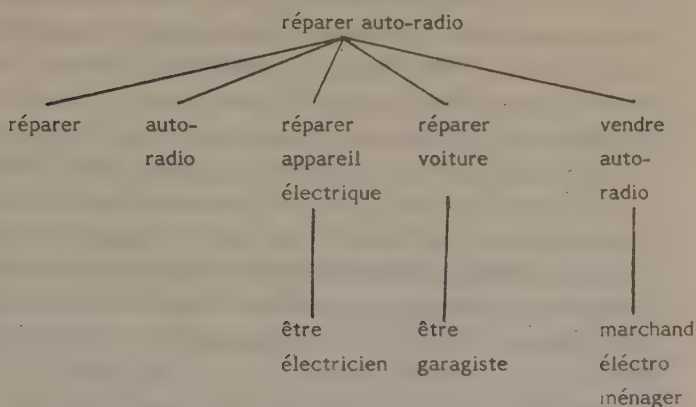


Fig III.2 - Arbre des thèmes

Les trois premières branches de IMEXEC traitent du même thème "réparer auto-radio", en partant des deux éléments de ce couple, mais ces recherches n'aboutissent pas. On passe alors au cas le plus intéressant dans la dernière branche, où le recours aux synonymies permet d'envisager des thèmes différents tels que "réparer voiture".

On utilise alors des noeuds de IMEXEC qui ne figurent pas en Fig II.6, pour des problèmes de place. On peut se reporter en Annexe pour trouver un résultat plus complet.

Notons au passage que cet arbre n'est pas rigoureusement identique à celui qui figure en Fig II.7, car nous n'avions alors reproduit cet arbre que partiellement : nous avons écarté, pour plus de clarté, les thèmes incomplets qui ne permettaient pas d'aboutir à des rubriques existantes.

III.2.2 La prise en compte des connaissances de l'utilisateur

Nous allons voir maintenant comment sont créées et modifiées les structures nous permettant de représenter les connaissances que nous attribuons à l'utilisateur, et le problème qu'il a posé.

III.2.2.1 Les connaissances sur le domaine d'application

Nous avons vu (§II.5.1) que nous supposons a priori certaines connaissances chez l'utilisateur, concernant le domaine sur lequel il expose sa requête. Ceci se traduit par l'attribution, au début de chaque dialogue, d'une liste de pointeurs vers des noeuds d'IMALGO, constituant la représentation des connaissances de ce nouvel utilisateur.

Dès que l'utilisateur formule une requête sur un sujet nouveau, RAISON est appelé et crée un arbre IMEXEC. Les pointeurs de cette liste sont alors transférés vers les noeuds correspondants dans IMEXEC. Plus précisément, une nouvelle liste de pointeurs est créée vers les noeuds de IMEXEC. Cette liste nous permet de connaître les noeuds de IMEXEC que nous devons utiliser afin de répondre à une éventuelle demande d'explication de l'utilisateur.

On obtient ainsi une représentation des connaissances attribuées à l'utilisateur sur le sujet particulier défini par la requête qu'il vient de présenter.

A chaque demande de l'utilisateur concernant ce même sujet, le module EXPLIQ prend en compte le même arbre IMEXEC pour formuler une explication (nous reviendrons plus en détail sur cette procédure dans le §III.3). La liste que nous avons constituée ci-dessus lui permet de déterminer les noeuds qu'il doit exploiter.

A la suite de chaque explication, nous modifions la liste de pointeurs, pour prendre en compte le fait que l'utilisateur semble avoir moins de connaissances sur le domaine que nous ne le supposons a priori. Les noeuds ayant été utilisés pour formuler l'explication sont alors remplacés par leurs descendants. Quand on ne peut plus descendre les pointeurs d'un niveau, c'est-à-dire lorsqu'ils pointent sur les feuilles de IMEXEC, on note alors sur la p-liste du pointeur, qu'une explication a déjà été fournie en exploitant ce noeud. Ceci montre que l'utilisateur n'a probablement pas les mêmes connaissances que le système en ce qui concerne cette donnée. En effet, lorsqu'on exploite les noeuds terminaux d'IMEXEC, ce ne sont plus les actions faites par RAISON qui sont mises en cause, mais les données utilisées par celui-ci.

Nous verrons plus loin comment EXPLIQ tente de prendre en compte ce phénomène pour donner une explication à l'utilisateur.

A chaque changement de sujet, on reprend la liste de pointeurs vers IMALGO fournie à l'initialisation du processus, pour créer une nouvelle liste de pointeurs vers les noeuds de l'arbre IMEXEC qui vient d'être créé. Nous ne transférons donc pas sur le nouveau sujet, les transformations produites sur le précédent.

Du fait du changement de domaine, nous avons considéré que la modification des connaissances de l'utilisateur sur un sujet, provoquée par une demande d'explication, n'influe pas obligatoirement sur ses connaissances sur un autre sujet.

Pour justifier ceci de manière plus profonde, nous aurions besoin de faire une étude statistique, afin de vérifier que le manque de connaissances, sur un domaine particulier, d'un utilisateur donné, doit être considéré comme un manque de connaissances sur tous les domaines, ou, au contraire, ne préjuge en rien de ses connaissances dans d'autres domaines.

Nous avons choisi la deuxième solution, pensant qu'il valait mieux rester plutôt concis dans une première explication, quitte à la détailler par la suite, que de se montrer trop "bavard" et perdre de la crédibilité vis à vis de l'utilisateur.

III.2.2.2 Définition du problème de l'utilisateur

Lorsque l'utilisateur expose sa requête, nous n'avons pris en compte, dans un premier temps, qu'une partie de celle-ci pour faire appel à RAISON : celle qui en constitue le thème. Les autres éléments constituant cette requête (quand ils existent, bien sûr) vont venir enrichir la représentation du problème de l'utilisateur, tel que le système peut se le représenter.

Tant qu'on reste dans un même sujet, à chaque intervention de l'utilisateur, on va indiquer dans cette représentation les éléments venant compléter le thème.

Nous aurons ainsi des précisions venant se greffer sur l'arbre des thèmes, qui permettent de cerner plus exactement le problème posé.

Prenons un exemple. Si l'utilisateur dit : "Je voudrais faire réparer mon auto-radio par un spécialiste", on crée alors une représentation de la partie "par un spécialiste" qui viendra pointer sur "réparer un auto-radio", en précisant : agent = spécialiste.

Dans la suite du dialogue, quand le système propose à l'utilisateur de choisir entre un garagiste, un vendeur d'auto-radio et un vendeur

d'électro-ménager, si l'utilisateur précise alors qu'il veut un garagiste dans le 5ème arrondissement, nous indiquerons un cas lieu associé au thème "être garagiste".

Ceci permet à DIALOG de lever certaines ambiguïtés, et de relancer le dialogue avec plus de précisions. Nous verrons cette utilisation dans le §III.4.

III.2.3) La constitution de l'historique

Rappelons la forme générale de l'historique du dialogue. A chaque intervention correspond une "page", identifiée par un numéro séquentiel. Le schéma en est le suivant :

I.i
Représentation interne de la phrase = - Intervention - Interlocuteur - Thème - Utilisateur - Système

Il contient un certain nombre de cases, contenant respectivement :

- un pointeur vers la RI de l'intervention;
- l'interlocuteur ayant prononcé l'intervention;
- un pointeur vers le thème de l'intervention;
- un pointeur vers la représentation de l'utilisateur telle qu'on la connaît au moment de l'intervention;
- un pointeur vers le raisonnement que le système a mené pour fournir sa réponse à l'utilisateur.

Ces différents éléments vont constituer des p-listes de l'atome V-Lisp représentant la page de l'historique.

Cette représentation est construite par DIALOG.

La première de ces cases est complétée aisément. Il s'agit en fait de la représentation issue de l'analyseur ou de celle qui est construite par DIALOG à partir des éléments obtenus par EXPLIQ ou RAISON.

La seconde identifie l'interlocuteur ayant prononcé l'intervention considérée. Il s'agit donc de l'utilisateur ou de la machine. Si l'intervention provient de l'analyseur, il s'agit de l'utilisateur; si elle a été constituée par DIALOG, il s'agit du système.

La case suivante contient le thème de l'intervention, ou plus précisément un pointeur vers un noeud d'un arbre de thèmes appartenant à la liste des différents sujets développés.

Quand il s'agit d'une intervention de l'utilisateur, la première action de DIALOG est d'en chercher le thème et d'essayer de l'attacher à un thème existant. Il sait alors où situer ce thème (dans un arbre existant, ou en créant un nouvel arbre). Dans les deux cas, on connaît alors l'emplacement du thème de l'intervention, et on peut remplir la case correspondante de l'historique.

Lors d'une intervention-système pour laquelle RAISON a été activé, nous avons vu que le module EXPLIQ constitue les descendants d'un arbre de thèmes dont la racine est la requête de l'utilisateur.

Nous pouvons alors retrouver dans cet arbre l'emplacement du thème de la réponse formulée par le système. Notons à ce sujet que si RAISON fournit plusieurs solutions et que DIALOG ne peut résoudre ces ambiguïtés, il peut arriver qu'il y ait plusieurs thèmes dans la réponse à donner.

C'est le cas dans l'exemple "Je veux réparer mon auto-radio", le système fournit le choix entre "être garagiste" et "être réparateur d'appareils électriques".

Dans l'historique nous ferons alors figurer ces deux thèmes.

La coexistence de plusieurs thèmes dans une intervention du système peut se produire également quand il fournit des explications. Nous connaissons l'arbre des thèmes concernés par l'explication, puisque nous recherchons le contexte de la demande d'explication, nous retrouvons ainsi aisément le ou les thèmes figurant dans la réponse fournie. L'historique peut être complété avec cette donnée.

La case correspondant à l'état de la représentation de l'utilisateur pointe toujours sur le dernier "état" de cette représentation. Il suffit de rechercher le dernier élément de la liste représentant ces différents états pour remplir l'historique avec un pointeur vers cet élément.

Le dernier élément composant une page de l'historique est un pointeur vers un raisonnement réalisé par RAISON. Il n'existe bien sûr que dans le cas où on enregistre une intervention du système pour laquelle RAISON a dû être activé.

Il pointe vers l'arbre IMEXEC qui vient d'être créé. Toutefois, nous remplissons également ce pointeur lorsque le système formule une explication. Il pointe alors vers l'arbre IMEXEC qui a été exploité pour la fournir.

Nous avons vu dans ce paragraphe comment les différentes représentations définies au chapitre II sont créées et mises à jour. Nous allons maintenant aborder les procédures d'explication et de relance du dialogue, qui vont utiliser ces représentations.

III.3 La procédure d'explication

Nous avons vu dans le §III.2.1 comment nous retrouvons le contexte dans lequel il faut situer une demande d'explication : en menant une recherche dans la liste des sujets abordés, puis en se reportant à l'historique.

Nous trouvons ainsi l'état des diverses représentations au moment où le thème sur lequel porte la demande a été introduit, et en particulier l'arbre IMEXEC créé pour répondre à la première requête de l'utilisateur sur ce sujet.

C'est à partir de cet arbre IMEXEC que le module EXPLIQ va réunir les éléments qu'il transmettra à DIALOG afin de répondre à l'utilisateur.

Plusieurs stratégies vont pouvoir être adoptées par EXPLIQ : elles sont fonction de la question de l'utilisateur. En effet, celui-ci peut ne mettre en cause qu'une partie des résultats qui lui ont été proposés, ou leur ensemble.

Voyons tout d'abord ce que fait EXPLIQ lorsque la demande ne porte que sur une partie de la réponse qui lui a été faite.

III.3.1 L'explication partielle

Dans ce cas, il nous faut déterminer quelle est la partie de l'arbre IMEXEC qui a permis d'aboutir au résultat mis en cause. Pour cela, nous devons d'abord retrouver un nœud de IMEXEC dans lequel figure le thème de la demande, ou le thème auquel nous avons pu attacher cette demande dans l'arbre des thèmes existants (cf §III.2.1.2.b)). Ce nœud n'est pas obligatoirement unique : nous avons vu, dans la construction d'un arbre de thèmes, qu'un même thème peut recouvrir plusieurs nœuds de IMEXEC. Il nous suffit de trouver l'un de ces nœuds, les autres étant alors ses descendants ou ses ascendants, et feront partie de la branche retenue.

Notons que ce nœud existera obligatoirement : si nous avons pu trouver dans l'arbre des thèmes un nœud auquel attacher la requête de l'utilisateur, il faut que ce nœud ait été créé à la suite de l'examen de l'arbre IMEXEC. Pour éviter le parcours de IMEXEC, nous envisageons d'ajouter, à chaque nœud de l'arbre des thèmes, un pointeur vers le nœud de IMEXEC où ce thème est abordé. Le défaut de cette méthode est d'alourdir la représentation des thèmes, mais nous pensons qu'elle serait préférable au parcours des nœuds de IMEXEC que nous devons effectuer actuellement.

Quand nous avons trouvé ce nœud, nous pouvons alors savoir quels sont les éléments qui composeront la réponse à formuler. Il suffit, pour cela, de considérer les nœuds figurant dans la même branche de IMEXEC, en prenant en compte tous ses descendants (dans la limite des connaissances de l'utilisateur), et ses ascendants dont le thème est identique à celui de la demande, ou peut s'y attacher.

Nous ne cherchons pas réellement à attacher les différents thèmes des nœuds de la branche de IMEXEC au thème de la demande. Nous prenons en compte les différents nœuds qui figurent dans l'arbre des thèmes, sur la branche correspondant au thème de la demande. Ce sont ces thèmes que nous allons comparer avec les noms des nœuds d'IMEXEC.

Si dans notre exemple : "Je voudrais réparer mon auto-radio", l'utilisateur demande pourquoi on lui propose un garagiste, nous examinons l'arbre des thèmes correspondants (voir Fig II.7), dans lequel nous sélectionnons les nœuds "être-garagiste" et "réparer-voiture". Nous éliminons le nœud racine "réparer auto-radio", puisque celui-ci ne nous permettrait pas de sélectionner les nœuds "intéressants" de IMEXEC.

Ainsi, EXPLIQ ne va retenir dans les antécédents du premier nœud de IMEXEC trouvé, que ceux dont le thème appartient à la liste constituée précédemment.

Soit : un auto-radio fait partie d'une voiture, voiture est un des noms de la rubrique garagiste et un garagiste répare une voiture. Les derniers nœuds sont supprimés en raison des connaissances supposées chez l'utilisateur (que l'on connaît par l'historique) et la réponse qui lui est donnée ne reprend que le lien découvert entre auto-radio et voiture.

Nous avons ainsi sélectionné les éléments devant constituer la réponse à fournir à l'utilisateur à la suite d'une demande ne concernant qu'une partie du raisonnement mené.

Voyons maintenant ce qui se passe quand la demande de l'utilisateur est plus générale.

III.3.2 L'explication générale

C'est le cas lorsque l'utilisateur émet des questions vagues ou elliptiques, telles que : "Pourquoi ?" ou "Pourquoi me dites-vous cela ?".

On se trouve alors face à des demandes qui peuvent être identifiées comme des demandes causales, mais dont le thème est

vide a priori : elles reprennent en fait celui de l'intervention précédente. Notons à ce sujet que les demandes de forme elliptique ne peuvent pas toujours être comprises par l'analyseur, et que l'intervention d'un dialogue devrait pouvoir aider à résoudre ce genre de problème.

Nous reviendrons dans notre conclusion sur la façon dont nous espérons pouvoir envisager ce genre de problèmes. Dans notre approche actuelle, nous considérons qu'ils sont résolus.

Il faut alors, pour formuler une explication, reprendre l'ensemble du travail réalisé par RAISON.

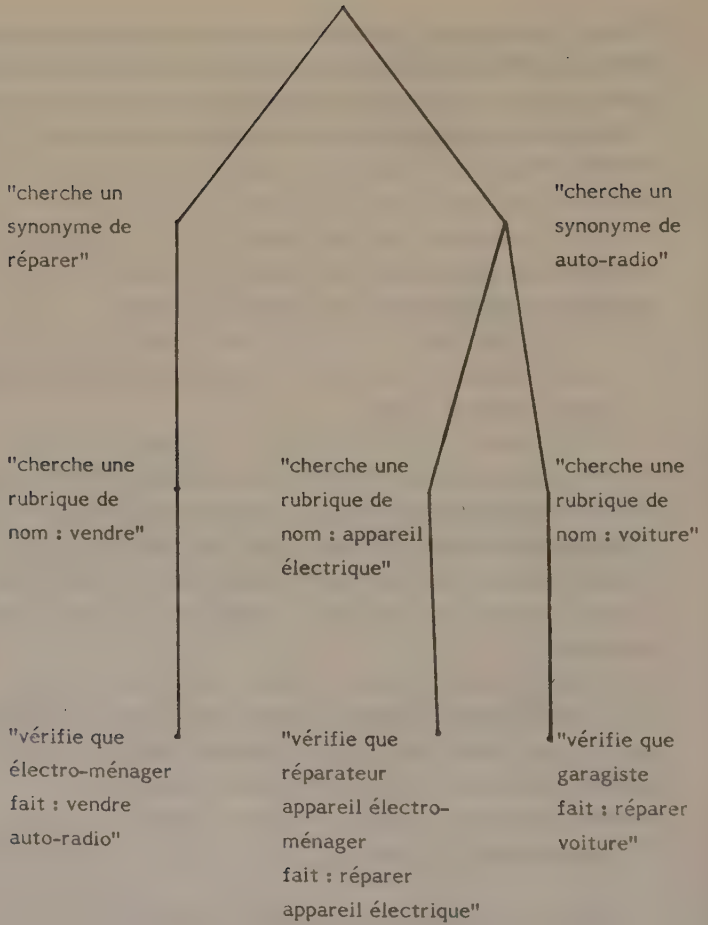
Pour cela, EXPLIQ va donc examiner l'ensemble des nœuds figurant dans IMEXEC, tout en tenant compte des connaissances qui sont supposées chez l'utilisateur à ce moment du dialogue. De ce fait, il faut également reprendre dans la représentation de l'utilisateur la liste des pointeurs vers cet arbre IMEXEC, que l'on trouve grâce à l'historique.

EXPLIQ sait alors qu'il doit formuler son explication en prenant dans chaque branche de IMEXEC les nœuds qui sont à un niveau supérieur à celui qui figure dans la liste des pointeurs.

Reprenons notre exemple.

Dans l'arbre IMEXEC obtenu à la suite de la requête "Je voudrais réparer mon auto-radio", les trois premières branches de IMEXEC mènent à des échecs : on ne retrouve pas dans les thèmes des différents nœuds, les thèmes ayant conduit à la réponse du système qui a provoqué une demande de l'utilisateur.

On va ensuite considérer la dernière branche. Dans celle-ci on trouve les nœuds suivants :



Les autres noms sont éliminés, car ils conduisent à des échecs, et n'ont pas figuré dans la réponse.

III.4 La relance du dialogue

Nous allons montrer ici comment nous avons utilisé les représentations précédentes afin de relancer le dialogue quand le module RAISON ne parvient pas à déterminer la solution au problème posé par l'utilisateur.

Ceci se produit en général quand le thème de la demande n'est pas assez précis. Nous considérons que les requêtes dont le contenu est contestable ont été éliminées. Ce travail fait partie de la recherche menée par J.B Berthelin (Sabah 84 bis).

RAISON a commencé une recherche afin d'essayer de répondre à l'utilisateur. Il a construit un arbre IMEXEC qu'il a transmis à EXPLIQ. DIALOG reçoit, pour sa part, un constat d'échec, prenant la forme d'une liste de rubriques, dont RAISON n'est pas sûr qu'elles répondent à la demande exprimée.

C'est là qu'intervient DIALOG. Il va chercher d'abord dans la représentation du problème posé par l'utilisateur (cf §II.5.2) s'il existe des éléments non encore utilisés qui permettraient de faire une sélection dans les rubriques. Ensuite, il essaie d'utiliser le travail fait par RAISON, (c'est-à-dire les différents chemins empruntés pour aboutir à cette liste), et les connaissances qu'il possède sur le domaine d'application (c'est-à-dire les informations sémantiques et pragmatiques du système) pour déterminer une "bonne" question à poser à l'utilisateur.

Détaillons cette démarche.

III.4.1 Comment utiliser la représentation du problème

La première étape de ce travail est de trouver la représentation du problème sur lequel on est en train de travailler.

Ce renseignement est retrouvé dans l'historique. On obtient ainsi l'arbre des thèmes correspondant au problème posé, et une liste de renseignements venant compléter ces thèmes. Cette liste peut être vide quand les questions de l'utilisateur sont très concises. DIALOG passe alors à l'étape suivante. (cf III.4.2).

Dans le cas contraire, la liste contient des informations figurant dans la question de l'utilisateur, qui n'ont pas été prises en compte : DIALOG essaie d'exploiter ces éléments afin de faire une sélection dans la liste des résultats partiels donnés par RAISON. Pour cela, on essaie de déterminer quels sont ceux qui permettent de satisfaire ce critère supplémentaire.

Notons que dans cette partie le fonctionnement de DIALOG est plus directement lié à l'application sur laquelle nous avons travaillé. Mais nous pensons qu'il pourrait être adapté à d'autres applications dont le principe reste assez proche.

Nous allons donc examiner, dans notre application, les informations pragmatiques qui définissent les rubriques afin de voir celles qui vont contenir le renseignement complétant la demande de l'utilisateur. On effectue ce travail pour chaque élément venant compléter la définition du problème.

Deux solutions sont possibles.

Examinons d'abord celle où aucune rubrique ne satisfait notre recherche. Dans cette hypothèse nous ne retenons pas l'élément supplémentaire comme étant significatif puisqu'il ne permet pas d'opérer une sélection dans notre liste. Nous préférons faire confiance à la recherche (même quand elle est approximative) effectuée par RAISON plutôt que de retenir ce qui n'était a priori qu'une information complémentaire dans la requête.

Dans le second cas, nous retrouvons l'élément qui complétait la requête de l'utilisateur dans certaines rubriques. Nous éliminons les autres : elles figurent toujours dans IMEXEC et dans l'arbre des thèmes, mais ne sont pas retenues pour formuler la réponse. Un ensemble de résultats est ainsi sélectionné. Il doit permettre de répondre plus précisément à la demande.

Prenons un exemple : "Je voudrais transporter des meubles par avion". Le thème retenu est "transporter des meubles" et la représentation du problème est constituée avec : "Instrument : avion". RAISON cherche un professionnel capable d'effectuer un transport de meubles. Il trouve plusieurs solutions :

- "transport par avion"
- "transport par bateau"
- "transport à l'étranger"
- "transport d'objets précieux".

Dans le nom (cf §II.2.2) de la rubrique "transport par avion", on retrouve l'instrument spécifié dans la représentation du problème. On sélectionne alors cette rubrique, et on élimine les autres.

Ces rubriques existent toujours en mémoire. De ce fait, si la première réponse faite à l'utilisateur ne le satisfait pas, il est toujours possible de lui en donner une autre, choisie parmi celles qui ont été trouvées par RAISON.

Quand cette procédure permet de réduire la liste des résultats à moins de trois éléments, ces résultats sont tous fournis à l'utilisateur afin qu'il fasse lui-même son choix. Notons que la limitation à trois éléments est arbitraire. Elle nous a semblé cohérente, mais nous ne pourrions la justifier que par des tests en vraie grandeur.

Dans le cas contraire, DIALOG passe à l'étape suivante, qui consiste à tenir compte du travail de IMEXEC et des connaissances du système sur le domaine pour déterminer une question à poser à l'utilisateur.

III.4.2 L'utilisation des autres représentations

Notre but, en utilisant les autres représentations (IMEXEC, l'arbre des thèmes qui en est issu, les connaissances sémantiques et pragmatiques sur le domaine) est de trouver une question à poser à l'utilisateur qui permette d'opérer une bonne partition dans la liste des résultats trouvés par RAISON.

En observant l'arbre des thèmes construit d'après IMEXEC, nous pouvons connaître le cheminement qui a permis d'aboutir aux différents résultats.

Plusieurs cas peuvent se produire :

- ces résultats peuvent provenir d'un même thème; ceci se produit par exemple dans la requête : "Je voudrais transporter des meubles". On trouve plusieurs rubriques (voir la liste précédemment citée), qui sont toutes issues du thème "transporter meubles". On ne peut pas se servir alors des connaissances sur le thème pour déterminer la question à poser.

- les résultats proviennent de déviations différentes à partir du thème initial; cela signifie qu'il existe plusieurs façons d'aborder le problème posé. Quand le nombre de ces déviations n'est pas trop important, on peut alors demander à l'utilisateur, quel thème semble le plus intéressant pour répondre à sa demande.

Reprenons l'exemple de "réparer un auto-radio".

L'arbre des thèmes nous donne trois solutions pour résoudre ce problème; on pourrait demander à l'utilisateur quelle est celle qui correspond le mieux à son problème : quelqu'un qui "répare une voiture", "répare un appareil électrique" ou "vend du matériel électro-ménager". Notons que cette solution n'est pas celle qui est retenue réellement dans cet exemple, dans la mesure où le nombre de rubriques n'était pas plus grand que le nombre de thèmes, et dans ce cas nous avons préféré présenter directement la liste des résultats à l'utilisateur.

Quand cette procédure ne permet pas de déterminer la question à poser à l'utilisateur, nous utilisons les connaissances sémantiques et pragmatiques sur le domaine d'application pour essayer de trouver un critère de séparation des rubriques.

Pour cela, nous examinons d'abord les "noms" des différentes rubriques, et nous cherchons ensuite si, dans le réseau sémantique, ces noms sont reliés, et la nature du lien quand il existe.

Prenons un exemple.

"Je voudrais transporter des objets", on sélectionne alors les rubriques :

- "transport par avion"
- "transport par bateau"
- "transport d'objets précieux"
- "transport d'objets lourds"
- "transport à l'étranger".

On trouve un lien entre avion et bateau, qui correspondent au moyen de transport, entre objets précieux et lourds, qui correspondent au type d'objet transporté.

Ces deux observations permettent de construire les deux types de questions suivantes :

- "Quel moyen de transport préférez-vous ?"

- "Quel type d'objets voulez-vous transporter ?"

Les questions sont ordonnées suivant le nombre décroissant de rubriques qu'elles concernent. Dès que l'utilisateur répond à l'une de ces questions, nous pouvons sélectionner une rubrique. Quand il ne souhaite pas faire de choix, nous passons à la question suivante.

Quand l'étude faite sur les noms des rubriques n'aboutit pas, nous effectuons le même type de travail sur les informations pragmatiques définissant les rubriques. Les recherches se font toujours sur les éléments venant compléter le thème qui a permis d'aboutir à la rubrique envisagée.

Dans l'exemple précédent, nous aurions examiné ce qui venait définir le transport dans les différentes rubriques. Il ne s'agit pas ici de prendre en compte des activités annexes ou parallèles d'un professionnel qui n'ont aucun rapport avec la raison qui nous a amené à étudier cette rubrique.

Remarquons ici que ce genre de procédure pourrait être appliqué pour trouver un numéro particulier à l'intérieur d'une rubrique, si les professionnels décrivaient individuellement leurs activités. Mais il faudrait aussi envisager des procédures plus complexes, ne serait-ce que pour interpréter des caractéristiques telles que "ouvert le soir jusqu'à 21h", ou des détails sur des activités particulières. Nous n'entrerons pas plus précisément dans ce sujet.

Nous avons simplement envisagé une liste de particularités permettant de "découper" les rubriques comprenant beaucoup de professionnels (comme garagiste, par exemple). Celles-ci concernent essentiellement, dans l'état actuel du système et en raison des contraintes précédemment citées, le quartier (nous inspirant ici de l'existence d'un annuaire par rues, qui semble assez utile dans une grande ville).

En conclusion, nous avons essayé de mettre en place diverses procédures devant permettre de résoudre le plus grand nombre de cas. Cette partie est plus directement liée à l'application, et nécessiterait une étude plus approfondie de ses caractéristiques pour être améliorée.

- CHAPITRE IV -

Quelques exemples de dialogues

Nous allons détailler sur trois exemples, l'évolution des représentations et le fonctionnement des procédures que nous avons définies dans les deux chapitres précédents.

On trouvera en annexe l'arbre IMALGO complet, construit à partir du programme RAISON de l'application "pages jaunes". Cet arbre nous servira pour les trois exemples, tous issus de cette application.

On trouvera également dans cette annexe, la représentation des connaissances que l'on suppose chez l'utilisateur au début du dialogue. Rappelons qu'elle est constituée d'une liste de pointeurs vers des nœuds de IMALGO.

Toujours en annexe, nous donnons le réseau sémantique utilisé pour ces exemples.

IV.1 Exemple 1

U1 : Je voudrais organiser une réunion.

S1 : Est-ce une réunion professionnelle, une conférence
ou une réception ?

U2 : Un congrès scientifique réunissant deux cent
personnes environ.

S2 : Je vous propose de consulter la rubrique :
organisation de congrès.

Le but de cet exemple est d'illustrer les principes de relance du dialogue.

Examinons en détail ce qui se passe lors de l'arrivée de la première requête de l'utilisateur.

La représentation interne de la phrase nous indique :

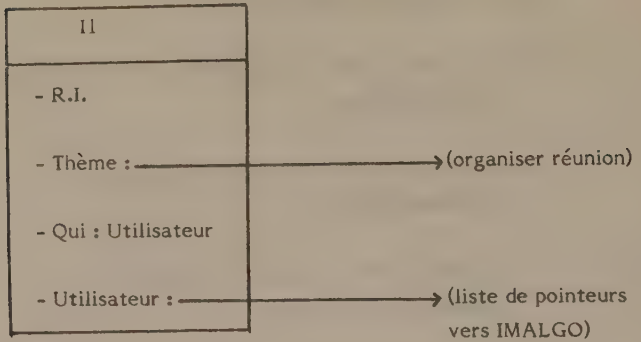
- le prédicat est : organiser

- le thème est : réunion.

La motivation nous permet de savoir qu'il s'agit d'une demande objet.

Le thème retenu est : (organiser réunion).

L'historique est constitué par DIALOG :

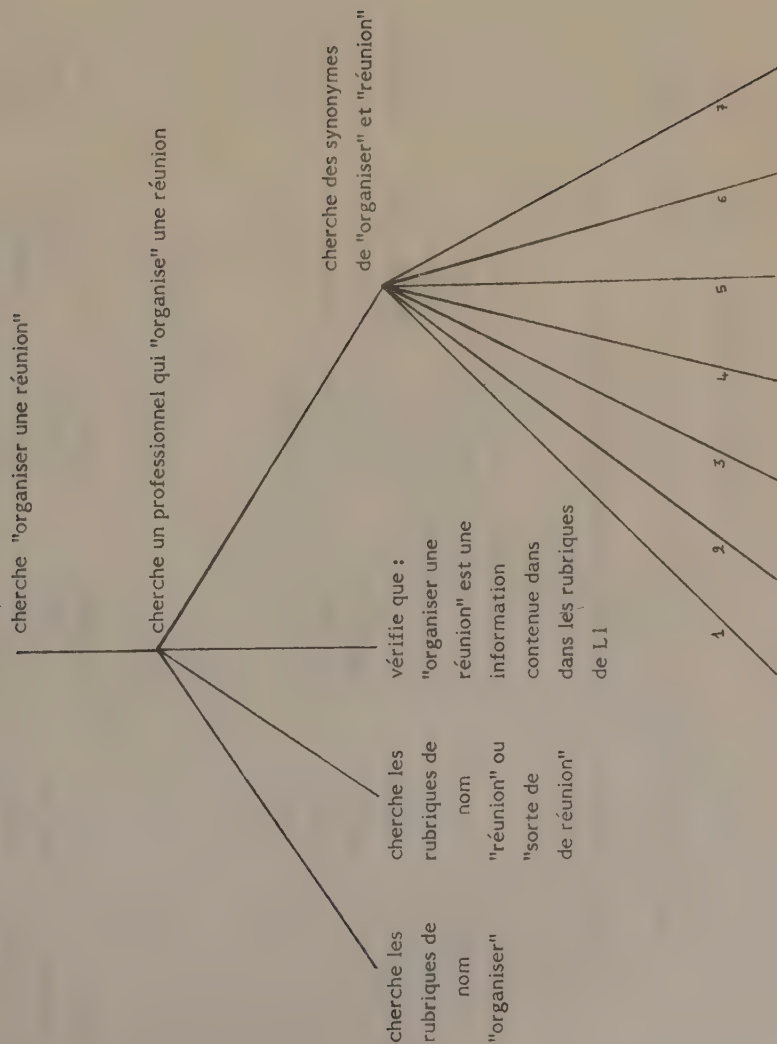


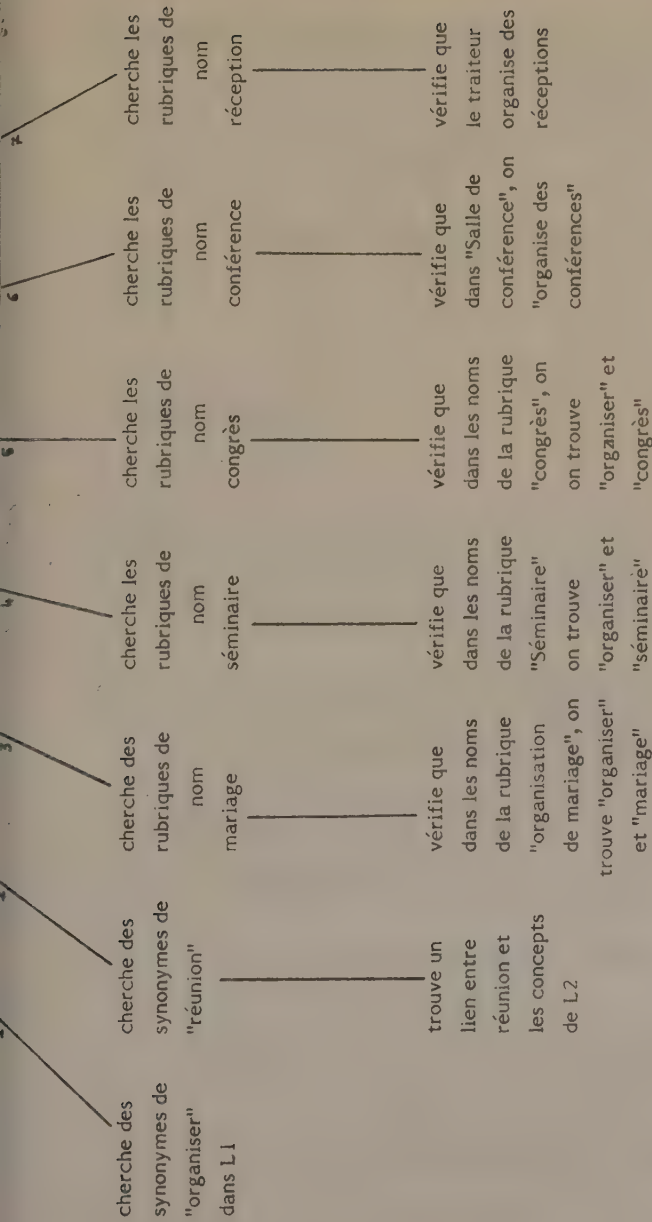
RAISON est activé avec ce thème.

Il ne trouve pas directement une rubrique où il est indiqué un professionnel qui organise des réunions. Grâce à des synonymies sur "réunion", il trouve les rubriques suivantes :

- Organisation de mariage
- Organisation de congrès
- Traiteur
- Séminaire
- Salle de conférence.

Il transmet cette liste à DIALOG, et envoie à EXPLIQ l'arbre IMEXEC1 traduisant ce fonctionnement.



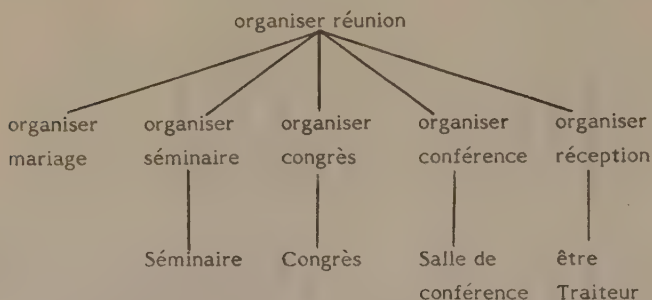


L1 : Liste des rubriques dont le nom est organiser, ou r union, ou "sorte-de" r union.

L2 : Liste des concepts  ventuellement li s   r union (c'est- -dire li    un  l ment qui lui-m me est li    r union).

Fig. IV.1 Arbre IMEXEC1

A partir de cet arbre IMEXEC1, on construit l'arbre des thèmes suivant :



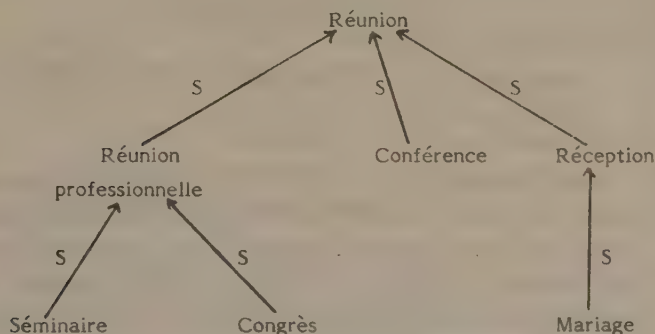
Le nombre de rubriques (5) est trop important pour que DIALOG en fournisse brutalement la liste. Il va donc essayer de trouver d'éventuelles inflexions dans la question de l'utilisateur : ici, il n'y en a pas, il passe donc à l'étape suivante.

Il observe quels sont les différents thèmes qui ont permis d'aboutir à ces rubriques. Ils sont également au nombre de 5, donc trop nombreux pour que DIALOG en fournisse la liste.

On effectue alors une étude sur les noms de rubriques, soit :

- organisation du mariage : organiser, mariage,...
- Séminaire : organiser, séminaire
- Congrès : organiser, congrès
- Salle de conférence : conférence
- Traiteur : organiser, réception, alimentation.

On trouve, dans le réseau sémantique, le schéma suivant, traduisant : un séminaire et un congrès sont des réunions professionnelles, un mariage est une réception, et, enfin, une réunion professionnelle, une réception et une conférence sont trois sortes de réunion.



(tous les liens S sont du type : "sorte-de")

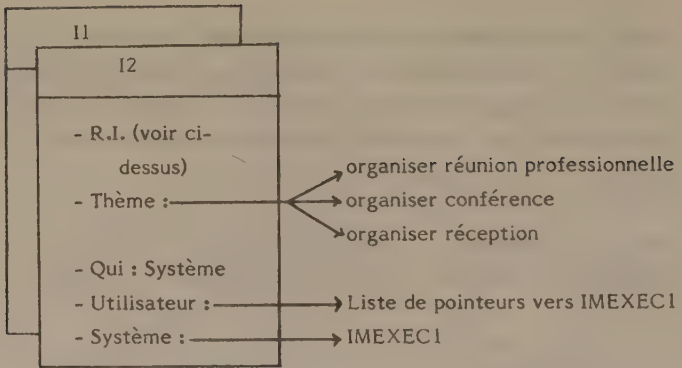
Fig. IV.2 Extrait du Réseau Sémantique

La structure de cet arbre nous induit à considérer les trois concepts Réunion professionnelle, Conférence, Réception qui recouvrent la sémantique des cinq solutions. On peut alors proposer le choix entre ces trois types de réunion, ce qui nous permettra de sélectionner selon la réponse une ou deux rubriques.

Afin de poser la question adéquate, DIALOG constitue donc une représentation contenant les informations :

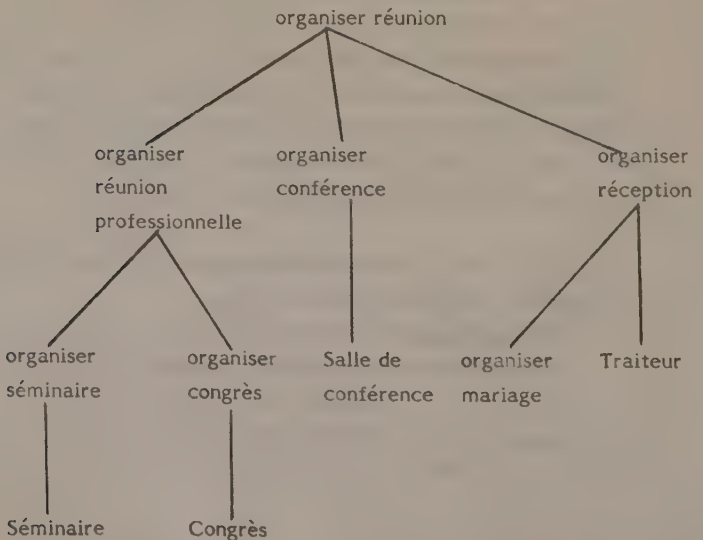
	Motivation → demande objet
	Objet : Réunion professionnelle, Conférence ou Réception

Il construit alors une nouvelle page de l'historique :



Deux remarques sont à faire sur ce schéma. La première concerne l'arbre des thèmes : du fait du lien entre séminaire et congrès, nous avons introduit un nouveau thème : "organiser réunion professionnelle". Il s'attache aux thèmes "organiser séminaire" et "organiser congrès". De façon analogue, "organiser mariage" s'attache à "organiser réception".

L'arbre des thèmes est donc modifié de la façon suivante :



La case "Thème" de la dernière page de l'historique (I2), pointe alors sur ce nouveau nœud, et sur les nœuds organiser réception et organiser conférence.

La seconde remarque concerne la représentation des connaissances de l'utilisateur. La liste des pointeurs vers IMALGO qui existait dans (I1), est transformée en liste de pointeurs vers l'arbre IMEXEC1 qui vient d'être créé par RAISON.

Après avoir constitué cette page de l'historique, DIALOG transmet la représentation interne construite au module de génération qui pose à l'utilisateur la question (S1).

L'utilisateur donne alors la réponse (U2), dont l'analyseur construit la représentation interne suivante :

Motivation→Information

Objet : Congrès infléchi par : - scientifique

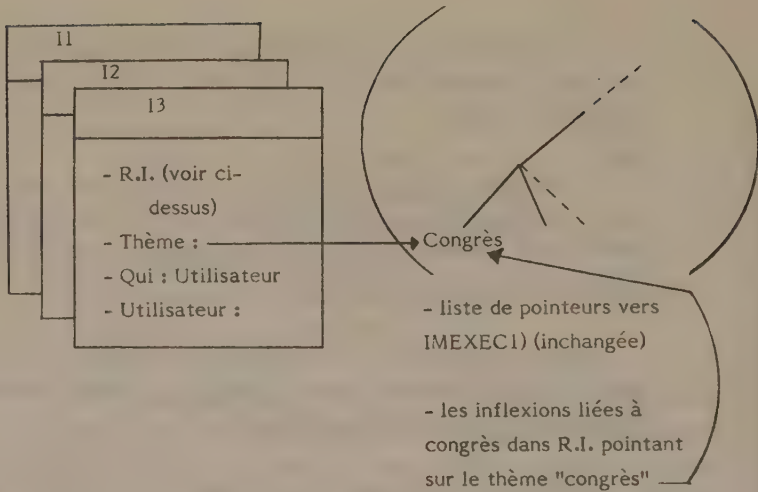
- assertion :

prédicat : réunir

objet : 200 personnes.

Le thème en est : Congrès.

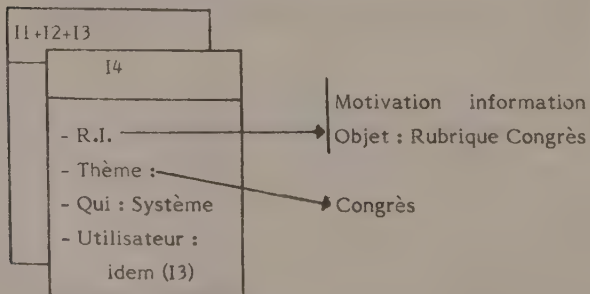
DIALOG reconnaît un thème déjà abordé, même s'il ne s'agit pas directement de l'un de ceux qui figurent dans la question précédente du système. Il constitue une nouvelle page de l'historique :



DIALOG ne fait pas appel à RAISON, puisqu'il reconnaît un thème déjà abordé. L'intervention de l'utilisateur est une information qui vient à la suite d'une demande du système (voir dans I2); de plus le thème de I3 est lié à l'un des thèmes de I2 : (U2) constitue donc une réponse à la question posée.

Cette information permet de sélectionner l'une des rubriques trouvées par RAISON : organiser un congrès. Cette rubrique permet ainsi de répondre à la requête initiale de l'utilisateur.

DIALOG construit alors la page I4 de l'historique :



Le module de génération pourra donner la réponse S2. Notons ici que nous n'avons pas essayé de choisir l'un des professionnels figurant dans cette rubrique. Les éléments figurant dans la représentation du problème de l'utilisateur, devraient permettre d'opérer une telle sélection, si nous pouvions connaître des renseignements concernant chaque professionnel, comme nous avons des informations pragmatiques sur les rubriques.

IV.2 Exemple 2

U1 : "Je voudrais transporter mon coffre-fort."

S1 : "Je vous propose un transporteur d'objet précieux ou un déménageur de piano."

U2 : "Pourquoi un déménageur de piano ?"

S2 : "Parce qu'un coffre-fort présente la même qualité qu'un piano : c'est un objet lourd."

U3 : "Pourquoi un transporteur d'objet précieux ?"

S3 : "Parce que votre coffre-fort peut contenir des objets précieux."

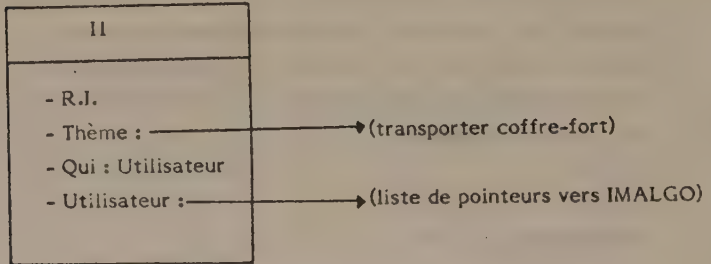
Le but de cet exemple est d'illustrer les procédures d'explication.

La première requête de l'utilisateur est analysée comme une demande objet avec :

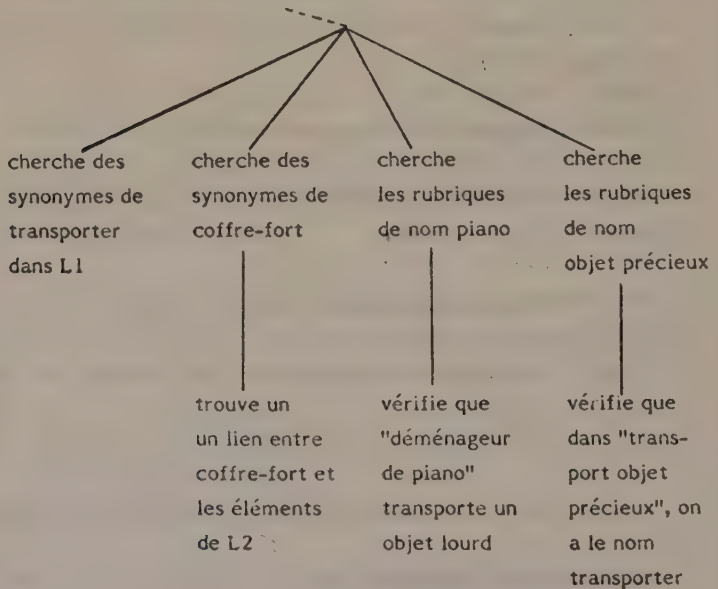
	Prédicat : transporter
	Objet : coffre-fort.

Le thème retenu est "transporter coffre-fort".

Comme dans l'exemple précédent, DIALOG construit la première page de l'historique :



RAISON est activé avec le thème (transporter un coffre-fort). Il construit l'arbre IMEXEC2 suivant :



L1 : liste des rubriques de nom transporter

L2 : liste de concepts ayant éventuellement un lien avec coffre-fort

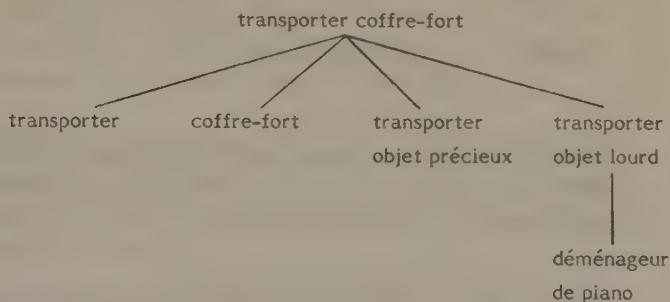
Fig. IV.3 Extrait de IMEXEC2

En résultat DIALOG reçoit les deux rubriques suivantes :

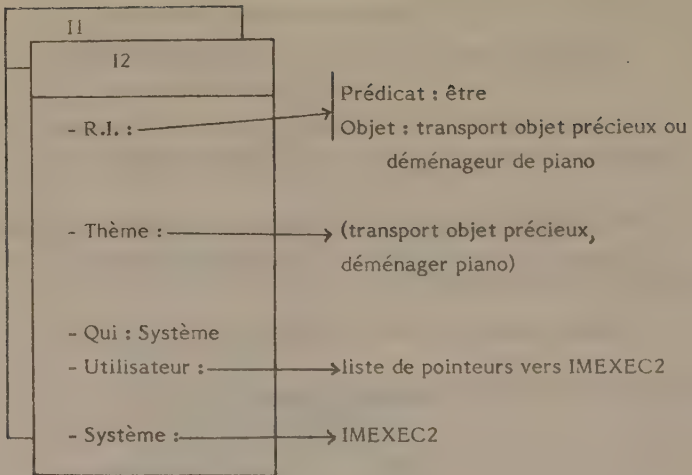
- déménageur de piano, obtenue par synonymie approchée entre coffre-fort et piano qui sont tous deux des objets lourds;

- transport objets précieux, obtenu par synonymie approchée entre coffre-fort et objet précieux : l'un peut contenir l'autre.

EXPLIQ complète à partir de IMEXEC2, l'arbre des thèmes :



DIALOG construit la représentation interne de la réponse contenant les deux rubriques trouvées. Il crée une nouvelle page de l'historique :



Le module de génération peut alors proposer l'intervention S1. L'utilisateur, surpris (on peut le comprendre !) par l'introduction d'un déménageur de piano, interroge le système (U2).

DIALOG reçoit la représentation interne de cette demande. Il reconnaît une demande causale, sur un sujet déjà abordé déménager piano. Il crée une nouvelle page de l'historique et fait appel à EXPLIQ pour retrouver les éléments permettant d'expliquer l'introduction de ce thème.

EXPLIQ consulte l'arbre IMEXEC2, trouve le nœud "cherche si déménageur piano transporte un objet lourd". Il sélectionne dans l'arbre des thèmes, les nœuds aboutissant à "déménager piano", soit : "transporter objet lourd". Il parcourt alors la branche de IMEXEC2 contenant le nœud cité plus haut. Il retient les nœuds dont le thème s'attache à "transporter objet lourd" ou "déménager piano", soit :

Le module de génération donne la réponse S2 à l'utilisateur. Celui-ci souhaite une nouvelle explication (U3), portant sur le deuxième élément de (S1).

DIALOG reçoit la représentation interne :

	Motivation→demande causale
	Prédicat : transporter
	Objet : objet précieux

DIALOG reconnaît un thème déjà abordé. Il sait qu'il s'agit d'une demande causale sur ce thème. Il fait appel à EXPLIQ afin d'obtenir les éléments de réponse, après avoir construit une page (I5) dans l'historique.

EXPLIQ effectue un travail analogue au précédent sur IMEXEC2. Il sélectionne les nœuds : "cherche un synonyme de transporter", "cherche un synonyme de coffre-fort", "trouve que coffre-fort peut contenir des objets précieux" (le nœud de vérification sur les noms des rubriques est considéré trop bas par la représentation des connaissances de l'utilisateur pour être exploiter).

Ces phrases sont transmises à DIALOG, qui modifie la représentation de l'utilisateur et construit une nouvelle page de l'historique (I6), analogue à (I4).

Le module de génération donne alors la réponse (S3).

IV.3 Exemple 3

U1 : "Je voudrais le numéro d'un restaurant d'extrême-orient."

S1 : "Préférez-vous vietnamien, japonais ou chinois ?"

U2 : "Cela m'est égal."

S2 : "Dans quel quartier voulez-vous aller ?"

U3 : "Je voudrai d'abord avoir le numéro de téléphone d'un taxi de nuit."

S3 : "Vous avez dans la rubrique taxi :

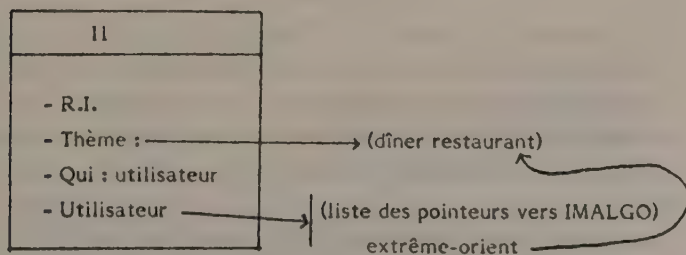
Allo Taxi (Tel : 111.11.11) et Olla Taxi (Tel : 222.22.22)."

U4 : "Dans ce cas je voudrais un restaurant dans le 5eme."

S4 : "Vous avez :

- restaurant chinois "le Mandarin" (333.33.33)
- restaurant japonais "le Soleil Levant" (111.11.11)."

La première requête de l'utilisateur est analysée, et nous indique qu'il s'agit d'une information interprétée comme une demande objet sur un restaurant d'extrême-orient. Une page de l'historique est constituée. Elle contient :



RAISON est activé avec le thème (dîner au restaurant). Il construit l'arbre IMEXEC3, dont voici le résumé :

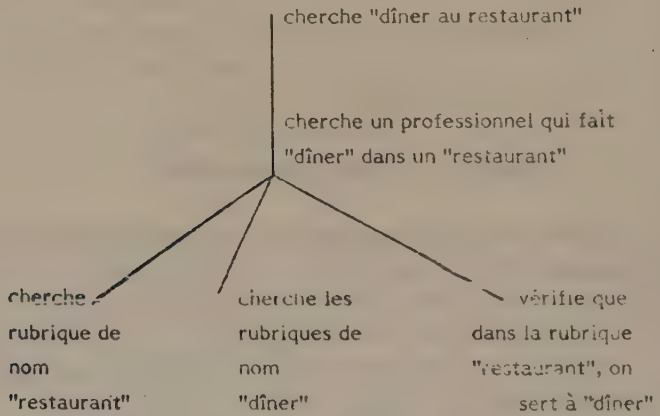


Fig. IV.4 IMEXEC3

RAISON trouve une liste de restaurants :

- restaurant chinois
- " vietnamien
- " brasserie
- " russe
- etc...

DIALOG reçoit cette liste. Il va d'abord consulter la représentation du problème de l'utilisateur, pour essayer de faire une sélection dans ces rubriques. Il trouve que restaurant est infléchi par extrême-orient. Il cherche dans les noms des rubriques obtenus par RAISON, lesquelles sont susceptibles de vérifier cette inflexion, en consultant les renseignements contenus dans le réseau sémantique. Il retient :

- restaurant chinois
- " vietnamien
- " japonais.

Il peut alors proposer à l'utilisateur (S1), après avoir constitué une page I2 dans l'historique.

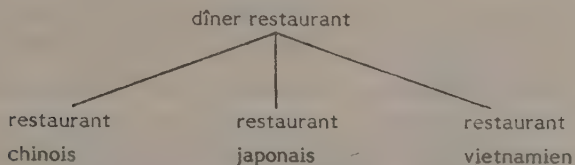
Dans (U2), l'utilisateur ne donne pas de réponse à cette question. Il faut donc que DIALOG propose une autre question. Pour cela, il effectue d'abord une recherche sur les thèmes ayant permis d'aboutir à ces rubriques : cette procédure ne donne rien, puisqu'il n'y a pas de différence entre ces thèmes.

Il consulte ensuite les noms et les informations pragmatiques des rubriques : il ne trouve pas d'autre question à poser que celle qu'il a déjà proposée.

Il arrive alors à la dernière solution, qui est de chercher dans les caractéristiques des rubriques, un critère de disjonction. Il pose alors la question sur le quartier (S2), après avoir construit une page (I4) de l'historique.

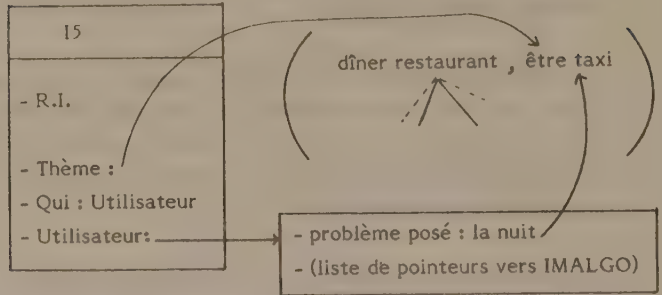
Dans (U3), l'utilisateur change de sujet : il passe à une autre demande.

DIALOG reçoit de l'analyseur, l'information qu'il s'agit d'une demande objet, portant sur un taxi de nuit. Il consulte la liste des thèmes déjà abordés, soit :



Il ne peut pas attacher ce nouveau thème, à un de ces nœuds. Il crée alors un nouvel arbre réduit à sa racine "être taxi".

Une nouvelle page de l'historique est créée :



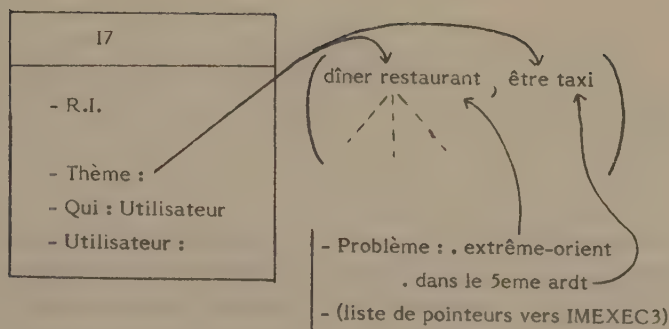
Notons ici, que dans la représentation des connaissances de l'utilisateur, nous avons repris la liste des pointeurs vers IMALGO, qui était attribuée au début du dialogue, et non pas celle qui existait dans 14 vers IMEXEC3, qui avait pu avoir été modifiée au cours du début du dialogue.

RAISON est activé avec le thème (être taxi). Il crée l'arbre IMEXEC4, et renvoie à DIALOG la rubrique taxi en résultat. Celui-ci peut alors vérifier que l'inflexion existant dans la requête, c'est-à-dire le fait que les taxis roulent la nuit, figure dans les informations pragmatiques de la rubrique (c'est ici qu'il faudrait ajouter une sélection à l'intérieur des professionnels de la rubrique, dans le cas où cette information ne serait pas vérifiée pour tous).

DIALOG construit une nouvelle page 16 dans l'historique, et donne au module de génération les éléments permettant de donner la réponse (S3).

L'utilisateur, muni de cette nouvelle information, demande alors un restaurant dans le 5ème arrondissement (U4).

DIALOG reçoit cette demande analysée. Il reconnaît un thème déjà abordé : celui de "restaurant"; il construit alors la page 17 suivante :



Ayant retrouvé le contexte de la demande, DIALOG rétablit la représentation des connaissances supposées chez l'utilisateur, c'est-à-dire la liste de pointeurs vers IMEXEC3, telle qu'elle était définie dans I4.

Il sait également qu'il ne s'agit pas d'une demande sur un nouveau sujet : RAISON n'est donc pas activé. Il cherche dans les données qu'il possédait au moment de la question (S2), quels sont les éléments qui vont pouvoir être sélectionnés, compte tenu de la nouvelle information concernant le quartier. Là encore, il a besoin de connaissances particulières sur les professionnels composant les trois rubriques : restaurant japonais, chinois et vietnamien pour sélectionner ceux qui répondent à la demande. Il peut alors transmettre au module de génération les éléments permettant de donner la réponse (S4).

Dans cet exemple, nous avons montré deux cas où une connaissance plus approfondie sur les données particulières du problème, permet de donner des réponses plus intéressantes. Dans notre application, nous avons mis en place une représentation embryonnaire de ces données afin de montrer comment elles pourraient être utilisées. Toutefois, comme nous l'avons déjà dit (§III.4.2), cette partie nécessiterait d'être approfondie pour être mieux utilisée, en employant en particulier des techniques de recherche documentaire.

- CONCLUSION -

Nous avons voulu montrer dans ce travail, l'importance du traitement du dialogue dans un système de questions-réponses. Quelles que soient les capacités de "raisonnement" d'un système, nous pensons qu'il est très difficile de convaincre un utilisateur non informaticien de son utilité, si sa façon de "dialoguer" avec lui ne présente pas un certain nombre de qualités.

Nous avons mis en évidence trois points qui nous ont semblé essentiels :

- la participation à un véritable dialogue;
- la "coopération" avec l'utilisateur afin de l'aider à préciser ses demandes;
- la "conscience" du travail réalisé en vue de l'expliquer.

Dans la plupart des systèmes de questions-réponses, le dialogue est généralement prévu à l'avance. L'interface se limite alors à faire le lien entre ce que dit l'utilisateur et ce que le système attend de lui. Le dialogue est dirigé par le système, plus ou moins ouvertement.

Notre but, en mettant en œuvre ces trois principes, est de rendre le contrôle du dialogue à l'utilisateur : lui permettre de changer de sujet ou de revenir en arrière, de poser des questions sur les réponses fournies par le système. Par ailleurs, quand le système ne parvient pas à résoudre complètement un problème, il faut qu'il apporte à l'utilisateur une aide efficace pour lui faire préciser sa demande. Pour cela, le système pose une question qui prend en compte le travail déjà effectué et qui indique à l'utilisateur les différents moyens de faire aboutir la recherche.

Pour réaliser ce travail, nous avons repris les études sur le dialogue humain. Nous ne prétendons pas avoir réussi à simuler un comportement humain dans n'importe quel type de conversation.

Constatant à quel point les systèmes censés "dialoguer" étaient loin de le faire réellement, nous avons voulu montrer qu'il fallait étudier les caractéristiques des conversations humaines pour obtenir des résultats plus satisfaisants.

Le problème est difficile : lors d'une conversation, chacun suppose une très grande quantité de connaissances chez son interlocuteur, tant sur le monde, que sur la façon normale de se comporter l'un vis à vis de l'autre dans un environnement donné. C'est ce que montre Grice dans ses maximes, où l'interprétation de la violation d'un des axiomes dérivés du principe de coopération, est aussi significatif que le contenu réel de l'intervention.

C'est pourquoi, nous avons construit notre système de façon à ce qu'il adopte un comportement respectant ces principes. Les interventions doivent être suffisamment claires pour que l'utilisateur n'ait pas à en chercher le sens caché.

Cette option nous semble justifiée dans le cadre d'un système de questions-réponses. Par contre, la réalisation de véritables dialogues, nécessiterait l'étude de ce que l'on pourrait appeler le "deuxième degré" d'interprétation des phrases prononcées, ce qui relève de la théorie de l'argumentation. Il faudrait être capable de construire et de comprendre des interventions dont le sens n'est pas seulement dans le contenu, mais essentiellement dans la façon de le présenter.

Ceci ne nous semble pas réalisable dans un avenir proche : les règles de comportement humain sont très difficiles à adapter pour obtenir d'une machine un résultat cohérent.

Le point qui reste le moins précis dans notre système, concerne la représentation que le système se fait de l'utilisateur. Du fait du caractère grand public de notre application, nous n'avons pas pu mettre en place une représentation précise des connaissances d'un utilisateur particulier. Il faudrait envisager d'adapter les réponses du système au niveau de chacun, plutôt que de n'avoir qu'une représentation a priori. Un axe de recherche serait alors d'étudier les termes employés dans la question, pour en apprécier le degré de précision.

Nous avons étudié les problèmes du dialogue, indépendamment des problèmes d'analyse des phrases.

Cette distinction est artificielle, dans la mesure où le dialogue devrait intervenir également dans ce processus, afin de demander des informations complémentaires lorsque l'analyseur ne peut pas interpréter les phrases de l'utilisateur. Nous pensons qu'il faudrait ici utiliser des procédures analogues à celles qui sont appliquées dans notre système, en cas d'échec (relatif) de la recherche dans la base de connaissances.

Il en est de même pour les problèmes d'apprentissage de concepts. Ces procédures devraient permettre d'exploiter les résultats partiels de l'analyseur et, ainsi, aider à la mise en œuvre d'un processus d'apprentissage.

C'est pourquoi, nous pensons que les procédures de dialogue et d'analyse doivent interagir. Le module de DIALOG ne doit plus se limiter à un "droit de regard" sur le module de raisonnement, il doit posséder la même capacité sur l'analyseur.

Nous espérons pouvoir réaliser alors un système plus complet, qui pourrait être utilisé dans d'autres applications.

PLAN DES ANNEXES

- 1°) Extrait du réseau sémantique
- 2°) Liste des rubriques
- 3°) Arbre Imaigo
- 4°) Structure d'un nœud d'Imexec

1°) Extrait du réseau sémantique.

Il s'agit ici des concepts utiles pour répondre aux requêtes présentées dans le chapitre IV.

On y trouve, pour chaque concept, la liste des concepts qui lui sont reliés avec le type du lien qui les unit.

(CONCEPT EXTREME-ORIENT (PARTIE-DE SORTE-DE) (SORTE-DE (CONTREE)
PARTIE-DE (ASIE)))
(CONCEPT VIETNAMIEN (QUALIF) (QUALIF (VIETNAM)))
(CONCEPT JAPONAIS (QUALIF) (QUALIF (JAPON)))
(CONCEPT CHINOIS (QUALIF) (QUALIF (CHINE)))
(CONCEPT RUSSE (QUALIF) (QUALIF (RUSSIE RUSSIE)))
(CONCEPT PROFESSION NIL NIL T)
(CONCEPT DERATISATION (SORTE-DE) (SORTE-DE (PROFESSION)))
(CONCEPT VENDRE NIL NIL)
(CONCEPT ACHETER NIL NIL)
(CONCEPT SCIGNER NIL NIL)
(CONCEPT LIEU NIL NIL T)
(CONCEPT ANIME NIL NIL T)
(CONCEPT ANIMAL (SORTE-DE) (SORTE-DE (ANIME)) T)
(CONCEPT INSECTE (SORTE-DE) (SORTE-DE (ANIMAL)))
(CONCEPT VERMINE (SORTE-DE) (SORTE-DE (ANIME)))
(CONCEPT CAFARD (SORTE-DE) (SORTE-DE (INSECTE VERMIN)))
(CONCEPT GARAGISTE (SORTE-DE) (SORTE-DE (PROFESSIONNEL)))
(CONCEPT CAOUTCHOUC (SORTE-DE) (SORTE-DE (MATIERE))) T
(CONCEPT ROUE (PARTIE-DE SORTE-DE FAIT-EN) (FAIT-EN (CAOUTCHOUC)
SORTE-DE (INANIME) PARTIE-DE (VEHICULE)))
(CONCEPT AUTOMOBILE (FCT-A SORTE-DE) (SORTE-DE (VEHICULE) FCT-A (ESSENCE)))
(CONCEPT VEHICULE (SORTE-DE) (SORTE-DE (INANIME)) T)
(CONCEPT AUTO-RADIO (PARTIE-DE FCT-A SORTE-DE) (SORTE-DE (PIECE-DET APP-ELEC) FCT-A (ELECTRICITE) PARTIE-DE (AUTOMOBILE)))
(CONCEPT INANIME NIL NIL T)
(CONCEPT APP-ELEC (SORTE-DE FCT-A) (FCT-A (ELECTRICITE) SORTE-DE (INANIME)) T)
(CONCEPT ESSENCE (SORTE-DE) (SORTE-DE (CARBURANT)))
(CONCEPT CARBURANT (SORTE-DE) (SORTE-DE (ENERGIE)) T)
(CONCEPT ELECTRICITE (SORTE-DE) (SORTE-DE (ENERGIE)) T)
(CONCEPT ENERGIE NIL NIL T)
(CONCEPT VOITURE (SORTE-DE) (SORTE-DE (AUTOMOBILE)))
(CONCEPT HOMME (SORTE-DE) (SORTE-DE (VIVANT)) T)
(CONCEPT PROFESSIONNEL (SORTE-DE) (SORTE-DE (HOMME)))
(CONCEPT DENT (PARTIE-DE) (PARTIE-DE (HOMME)))
(CONCEPT LUNETTES (SORTE-DE) (SORTE-DE (INANIME)))
(CONCEPT GARAGE (SORTE-DE) (SORTE-DE (LOCAL-PROF)))
(CONCEPT LOCAL-PROF (SORTE-DE) (SORTE-DE (LIEU)) T)
(CONCEPT REPARER NIL NIL)
(CONCEPT TUER NIL NIL)
(CONCEPT CONSTRUIRE NIL NIL)
(CONCEPT DEBARRASSER NIL NIL)
(CONCEPT CONGRES (SORTE-DE SORTE-DE) (SORTE-DE (REUNION-PROF) SORTE-DE NIL))
(CONCEPT REUNION-PROF (SORTE-DE) (SORTE-DE (REUNION)))
(CONCEPT REUNION NIL NIL)
(CONCEPT RECEPTION (SORTE-DE) (SORTE-DE (REUNION)))
(CONCEPT MARIAGE (SORTE-DE) (SORTE-DE (RECEPTION)))
(CONCEPT SEMINAIRE (SORTE-DE) (SORTE-DE (REUNION-PROF)))
(CONCEPT TAXI (SORTE-DE) (SORTE-DE (MOYEN-TRANSP VOITURE)))
(CONCEPT MOYEN-TRANSP NIL NIL T)
(CONCEPT RESTAURANT (SORTE-DE) (SORTE-DE (LOCAL-PROF)))
(CONCEPT TRANSPORTEE NIL NIL)
(CONCEPT OBJET (SORTE-DE) (SORTE-DE (INANIME)))
(CONCEPT LOURD (SORTE-DE) (SORTE-DE (QUALITE)))
(CONCEPT PLANO (QUALITE SORTE-DE) (SORTE-DE (OBJET) QUALITE (LOURD)))
(CONCEPT COFFRE-FORT (QUALITE SORTE-DE) (SORTE-DE (OBJET) QUALITE (LOURD)))
(CONCEPT DERATISATION (SORTE-DE) (SORTE-DE (PROFESSION)))
(CONCEPT DERNIERE NIL NIL)

2°) Liste des rubriques.

Nous donnons ici la liste des rubriques parmi lesquelles se trouvent celles qui permettent de répondre aux requêtes données en exemple dans ce travail.

On y retrouve l'intitulé de la rubrique accompagné de la liste de ses noms (le nombre de ces derniers variant entre 1 et 3°, et de la liste des phrases décrivant l'activité des professionnels. Ces dernières se présentent sous forme de représentation interne.

(RUBRIQUE ORG-MARIAGE (MARIAGE ORGANISER) ((PI17 ORGANISER (THEME)
MARIAGE) (PI18 VENDRE (THEME) ROBES)))
(RUBRIQUE AUTO-ECOLE (AUTO-ECOLE CONDUIRE ENSEIGNER) ((PI20
ENSEIGNER (THEME AGENT) (SPI5 CONDUIRE (THEME) AUTOMOBILE) MONITEUR)

))

(RUBRIQUE DERATISATION (DERATISATION) ((PI11 DERATISER (THEME)
MAISON) (PI12 TUER (THEME) VERMINE) (PI14 DEBARRASSER (THEME).
INSECTE) (PI15 DEBARRASSER (THEME) VERMINE)))

(RUBRIQUE DENTISTE (DENT DENTISTE) ((PI3 REPARER (THEME AGENT)
DENT DENTISTE) (PI4 SOIGNER (THEME AGENT) DENT DENTISTE)))

(RUBRIQUE OPTICIEN (OPTICIEN) ((PI1 REPARER (THEME AGENT) LUNETTES
OPTICIEN) (PI2 VENDRE (THEME AGENT) LUNETTES OPTICIEN)))

(RUBRIQUE GARAGE-AUTO (GARAGISTE AUTOMOBILE GARAGE) ((PI9 REPARER (THEME AGENT) VOITURE GARAGISTE)))

(RUBRIQUE ELECTRICITE (ELECTRICITE) ((PI10 REPARER (THEME AGENT)
APP-ELEC ELECTRICIEN)))

(RUBRIQUE CONSTR-CARAGE (GARAGE CONSTRUIRE) ((PI13 CONSTRUIRE (THEME) GARAGE)))

(RUBRIQUE CONGRES (CONGRES ORGANISER) ((PI47 ORGANISER (THEME)
CONGRES) (PI48 ENGAGER (THEME) INTERPRETE) (PI49 LOUER (THEME)
SALLE) (PI50 ORGANISER (THEME) SEJOUR)))

(RUBRIQUE SEMINAIRE (SEMINAIRE ORGANISER) ((PI44 ORGANISER (THEME)
SEMINAIRE) (PI45 ENGAGER (THEME) INTERPRETE) (PI46 LOUER (THEME)
LOCAL)))

(RUBRIQUE TRAITEUR (ALIMENTAIRE RECEPTION ORGANISER) ((PI51
PREPARER (THEME) PLAT) (PI52 LOUER (THEME) VAISSELLE) (PI53 FOURNIR
(THEME) SERVEUR) (PI54 INSTALLER (THEME) DECORATION)))

(RUBRIQUE SALLE-DE-CONF (CONFERELECT) ((PI41 LOUER (THEME) SALLE) (PI42 ENGAGER (THEME) INTERPRETE) (PI43 ENGAGER (THEME) CONFERENCEUR)))

(RUBRIQUE TR-OBJ-PREC (OBJ-PREC TRANSPORT) ((PI30 PROFITER (THEME)
MARCHANDISE)))

(RUBRIQUE TR-AVION (AVION TRANSPORT) ((PI38 ENREGISTRER (THEME)
FRET)))

(RUBRIQUE TR-BATEAU (BATEAU TRANSPORT) ((PI36 EMBARQUER (THEME)
MARCHANDISES) (PI37 DEBARQUER (THEME) MARCHANDISE)))

(RUBRIQUE TR-ETRANG (ETRANGER TRANSPORT) ((PI33 ACCOMPLIR (THEME)
DEDOUANAGE) (PI34 TRANSPORTER (THEME) MARCHANDISE) (PI35 FRANCHIR (THEME) FRONTIERE)))

(RUBRIQUE DEM-PIANO (PIANO DEMENAGER) ((PI31 DEMENAGER (THEME AGENT) PIANO TRANSPORTEUR)))

(RUBRIQUE REST-CHINOIS (CHINOIS RESTAURANT) ((PI29 BOIRE (THEME)
THEO) (PI30 MANGER (INSTR) BAGUETTE)))

(RUBRIQUE REST-JAP (JAPONAIS RESTAURANT) ((PI27 MANGER (THEME)
SUCHI) (PI28 BOIRE (THEME) SAKE)))

(RUBRIQUE REST-VIET (VIETNAMIEN RESTAURANT) ((PI25 MANGER (THEME)
NEM) (PI26 BOIRE (THEME) VIN-ROSE)))

(RUBRIQUE REST-BRASS (BRASSERIE RESTAURANT) ((PI23 VENDRE (THEME)
BIERE-PRESSIO) (PI24 SERVIR (THEME) CHOUCRROUTE)))

(RUBRIQUE REST-RUSSE (RUSSE RESTAURANT) ((PI21 JOUER (INSTR THEME AGENT) VIOLON MUSIQUE VIOLONISTE) (PI22 BOIRE (THEME) VODKA)))

(RUBRIQUE PIZZERIA (PIZZA RESTAURANT) ((PI19 PREPARER (THEME AGENT)
PIZZAS CUISINIER) (PI20 PREPARER (THEME AGENT THEME AGENT)
PLAT-ITALIEN CUISINIER PLAT-ITALIEN CUISINIER)))

3°) L'arbre Imalgo.

Nous donnons ici l'arbre Imalgo, image du programme RAISON pour l'application sur les pages jaunes de l'annuaire téléphonique.

Pour chaque nœud, représentant une action du programme, on trouve son père, la liste de ses fils et son nom, c'est-à-dire l'analyse du commentaire décrivant la fonction.

```
(resul raistot)
NOEUD: N1
PERE: NO
FILS: (N12 N11 N13 N14 N15)
"nom:"
"    predicat:" CHERCHER
"    " THEME ":predicat:" REPONDRE
"    " THEME !Q
"    " AGENT RUBRIQUE
PROF: 1
NOEUD: N11
PERE: N1
FILS: NIL
"nom:"
"    predicat:" CHERCHER
"    " THEME ":predicat:" LIER
"    " THEME !C2
"    " THEME !C2
"    " AGENT !L
PROF: 2
NOEUD: N12
PERE: N1
FILS: NIL
"nom:"
"    predicat:" CHERCHER
"    " THEME ":predicat:" LIER
"    " THEME !C2
"    " THEME !C2
"    " AGENT !L
PROF: 2
NOEUD: N13
PERE: N1
FILS: NIL
"nom:"
"    predicat:" CHERCHER
"    " BUT ":predicat:" ETRE-NOM
"    " BUT RUBRIQUE
"    " THEME !M
PROF: 2
NOEUD: N14
PERE: N1
FILS: (N141)
"nom:"
"    predicat:" CHERCHER
"    " THEME ":predicat:" ASSOCIER
"    " THEME !C
"    " AGENT RUBRIQUE
PROF: 2
NOEUD: N141
PERE: N14
FILS: NIL
"nom:"
"    predicat:" VERIFIER
"    " THEME ":predicat:" APPARTENIR
"    " THEME !PI
"    " AGENT !C
PROF: 2
```

NOEUD: N15
 PERE: N1
 FILS: (N151 N153 N154)
 "nom:"
 " predicat:" CHERCHER
 " " THEME ":predicat:" IV
 " " THEME IO
 PROF: 2
 NOEUD: N151
 PERE: N15
 FILS: NIL
 "nom:"
 " predicat:" CHERCHER
 " " THEME ":predicat:" FAIRE
 " " THEME IV
 " " AGENT PROFESSIONNEL
 PROF: 3
 NOEUD: N152
 PERE: N15
 FILS: NIL
 "nom:"
 " predicat:" CHERCHER
 " " BUT ":predicat:" ETRE-NOM
 " " BUT RUBRIQUE
 " " THEME IM
 PROF: 3
 NOEUD: N153
 PERE: N15
 FILS: (N1521)
 "nom:"
 " predicat:" CHERCHER
 " " LIEN ": INFGS
 " " THEME ": IPH
 PROF: 3
 NOEUD: N1531
 PERE: N153
 FILS: NIL
 "nom:"
 " predicat:" COMPARER
 " " THEME ": IPH
 " " THEME ": IPH
 PROF: 4
 NOEUD: N154
 PERE: N15
 FILS: (N1542 N1543 N1543)
 "nom:"
 " predicat:" CHERCHER
 " " THEME ":predicat:" ETRE-SYNO
 " " THEME IO
 " " THEME IO
 PROF: 3
 NOEUD: N1541
 PERE: N154
 FILS: (N15411)
 "nom:"
 " predicat:" CHERCHER
 " " BUT ":predicat:" LIEN
 " " BUT CONCEPT
 " " THEME IO
 PROF: 4

NOEUD: N15411
PERE: N1541
FILS: (N154111)

"nom:"
" predicat:" CITE:
" " BUT ":predicat:" LIFE
" " BUT CONCEPT
" " THEME IO

PROF: 5
NOEUD: N154111
PERE: N15411
FILS: NIL

"nom:"
" predicat:" COMBINER
" " THEME ":predicat:" NIL
" " THEME ":predicat:" NIL

PROF: 6
NOEUD: N1542
PERE: N154
FILS: (N15421)

"nom:"
" predicat:" CHERCHER
" " BUT ":predicat:" ETRE-NOM
" " BUT RUBRIQUE
" " THEME IM

PROF: 4
NOEUD: N15421
PERE: N1542
FILS: (N154212 N1531)

"nom:"
" predicat:" CHERCHER
" " LIEU ": " INTOS
" " THEME ": " IPH

PROF: 5
NOEUD: N1542
PERE: N154
FILS: (N15431 N15432)

"nom:"
" predicat:" CHERCHER
" " LIEU ":predicat:" ETRE-AUTRE
" " LIEU ": " V
" " THEME IV

PROF: 4
NOEUD: N15431
PERE: N1543
FILS: NIL

"nom:"
" predicat:" CHERCHER
" " THEME ":predicat:" FAIRE
" " THEME IV
" " AGENT PROFESSIONNEL

PROF: 5
NOEUD: N15432
PERE: N1543
FILS: (N154321)

"nom:"
" predicat:" CHERCHER
" " THEME ": " IV

PROF: 5

NOUE: N154321 ,
PERE: N15432
FILS: NIL
"nom:"
" predicat:" CHERCHER
" " LIEU ":" IP
" " THEME ":" OBJET
PROF: 6
NOEUD: N154212
PERE: N15421
FILS: (N1531)
"nom:"
" predicat:" CHERCHER
" " LIEU ":" INFOS
" " THEME ":" IPH
PROF: 6
FIN
FIN

4°) La structure d'un nœud de Imexec.

Un nœud d'un arbre IMEXEC se présente sous la forme suivante:

- Identification : 15N1531, par exemple.

La deuxième partie de cette identification indique l'identification du nœud de Imalgo correspondant (soit N1531). Le nombre qui le précède indique que l'on a déjà effectuée 14 fois l'action spécifiée dans ce nœud dans l'exécution particulière symbolisée par cet arbre Imexec.

- Le père de ce nœud.

Il s'agit d'une des occurrences du nœud père de N1531 de Imalgo.

- La liste des fils du nœud.

Il s'agit d'une liste d'occurrences de certains des nœuds fils de N1531 dans Imalgo.

- Le nom du nœud.

C'est la représentation interne figurant le nom de N1531 dans Imalgo, où les variables sont remplacées par leur valeur à cet instant de l'exécution.

BIBLIOGRAPHIE

- (Adam 78) : Adam A.
Utilisation des transformations sémantiques pour la correction
automatique de programmes.
Thèse d'Etat, Université Paris 6.
- (Allen 80) : Allen J.F. et Perrault C.R.
Analyzing intention in Utterances.
Artificial Intelligence, n°15, p.143-178.
- (Austin 62) : Austin J.L.
How To Do Things With Words
Oxford University Press, New-York.
- (Berthelin 84) : Berthelin J.B. et Rady M.
Déterminisme et Contestation sont dans un bateau.
Colloque Arc-84, Orsay.
- (Bien 83) : Bien J. et Wilks Y.
Beliefs, Points of view and Multiple environments.
Cognitive Science, Vol.7 n°2.
- (Bobrow 77) : Bobrow D.G., Kaplan R.M., Norman D.A., Thomson H.,
Winograd T.
GUS : A Frame-Driven Dialogue System,
Artificial Intelligence, Vol n°8, p.155-173.
- (Boden 77) : Boden M.
Artificial Intelligence and Natural Man.
Harvester Press, Hassocks.
- (Bonnet 79) : Bonnet A.
Understanding medical jargon as if it were a natural language.
6th IJCAI, Tokyo.
- (Borillo 77) : Borillo A.
Représentation des connaissances : de l'analyse syntaxique
de la phrase à son interprétation.
Journées de travail sur la compréhension, IRIA Sesori.
- (Bullwinckle 77) : Bullwinckle C. (voir aussi Sidner)
Levels of complexity in discourse for anaphora disambiguation
and speech act interpretation.
6th IJCAI, Tokyo.
- (Clark 82) : Clark H.H. et Murphy G.L.
La visée vers l'auditoire dans la signification et la référence.
Bulletin de psychologie n°356, Tome XXXV, Paris, p.767-

- (Dreyfus 79) : Dreyfus H.L.
What computers can't do
Harper.
- (Ducrot 72) : Ducrot O.
Dire et ne pas dire. Principes de sémantique linguistique.
Hermann, Savoir, Paris.
- (Ferber 83) : Ferber J.
MERING : un langage d'acteur pour la représentation et la
manipulation de connaissances.
Thèse de 3ème cycle, Université Paris 6.
- (Fillmore 68) : Fillmore C.J.
The Case for Case
In Universals in Linguistic Theory, Ed by Bach&Harms,
Holt Rinehart&Winston Inc., p.1-90.
- (Fouet 82) : Fouet J.M.
Système de compréhension de programmes.
Globule, Décembre 82.
- (Goldstein 79) : Goldstein I. et Roberts B.
Using frames in scheduling.
In A.I., a MIT Perspective, Vol 1, eds P.H.Winston and C.H.Brown
the MIT Press, Cambridge.
- (Gau 83) : Gau B.
Analyse et Représentation d'un texte d'après le thème du
discours,
Thèse de Troisième Cycle, Paris 6.
- (Grice 75) : Grice H.P.
Logic and Conversation
In Cole P. and Morgan J.L., Ed Syntax and Semantics,
Speech Acts, Academic Press, New-York.
- (Grosz 77) : Grosz B.
The representation and use of focus in understanding,
Standford Research Institute, Technical Note 151,
Menlo Park, California.
- (Halliday 61) : Halliday M.A.K.
Categories of the theory of grammar
Word 17, p.241-292.
- (Hayes 79) : Hayes P. et Reddy R.
An anatomy of graceful interaction in spoken and written
Man-Machine Communication.
Dpt of Computer Science, Carnegie Mellon University.

- (Kayser 84) : Kayser D.
Examen de diverses méthodes utilisées en Représentation
des connaissances.
4ème Congrès AFCET RF et IA, Tome 2, Paris, p.115-144.
- (Langlotz 83) : Langlotz C.P., Shortliffe E.H.
Adapting a consultation system to critique user plans
In Int.J.Man-Machine Studies, 19, p.479-496.
- (Laurent 82) : Laurent J.P. et Fouet J.M.
Outils pour la manipulation de programmes.
1er Congrès de génie logiciel, Paris.
- (Lehnert 77) : Lehnert W.
Human and computational question-answering.
Cognitive Science I (1).
- (Lehnert 83) : Lehnert W.G., Dyer M.G., Johnson P.N., Yang C.J., Harley S.
Boris : An experiment in in-depth understanding of narratives
Artificial Intelligence, Vol n°20-1.
- (Minsky 74) : Minsky M.
A framework for representing knowledge.
Memo 306, MIT Cambridge.
- (Perrault 78) : Perrault C.R., Allen J.F. et Cohen P.R.
Speech acts as a basis for understanding dialogue coherence
in Proceedings of 2nd Conference on Theoretical Issues in
Natural Processing, Champaign-Urbana.
- (Pitrat 79) : Pitrat J.
Réalisation d'un analyseur-générateur lexicographique général.
Rapport 79-2 du GR22, Université Paris 6.
- (Rady 83) : Rady M.
L'ambiguïté du langage naturel est-elle la source du
non-déterminisme des procédures de traitement ?
Thèse d'état, Université Paris 6.
- (Raphael 64) : Raphael B.
A computer program which "understands".
AFIPS Vol 26 Part I, p. 577-589.
- (Reichmann 78) : Reichmann R.
Conversational Coherency
Cognitive Science n°2.
- (Revenu 82) : Revenu M.
Mise sous d'arborescence syntaxique d'un programme écrit en
Pascal à l'aide d'un système-expert.
Publications GR22 n°30, Le Mans, p.283-302.

- (Sabah 76) : Sabah G.
Contribution à la compréhension d'un récit
Thèse d'état, Université Paris 6.
- (Sabah 83) : Sabah G.
Un système de questions-réponses sur les rubriques
professionnelles de l'annuaire téléphonique.
Rapport ATP, Publications CNRS GR22, n°31.
- (Sabah 84) : Sabah G.
Quelques notions de synonymies approchées.
Colloque Arc-84, Orsay.
- (Sabah 84 bis) : Sabah G., Vilnat A., Berthelin J.B.
Le dialogue dans un système de questions-réponses.
Séminaire "Dialogue homme-machine à composante orale".
Greco, Nancy.
- (Sabatier 83) : Sabatier P.
Vers des interfaces en langage naturel transparentes.
Congrès AFCET IA et RF, Paris.
- (Sadock 77) : Sadock J.
Modus brevis : the truncated argument
Papers from the 13th Regional Meeting Chicago Linguistics
Society, Chicago, Illinois.
- (Schank 77) : Schank R.C. et Abelson R.P.
Scripts, plans, goals and understanding.
Lawrence Erlbaum Press, Hillsdale, N.J.
- (Schank 79) : Schank R., Lehnert W.G.
The conceptual content of conversation
In IJCAI 1979, Tokyo, p.769-771.
- (Searle 79) : Searle J.R.
Expression and Meaning
Cambridge University Press.
- (Shortliffe 76) : Shortliffe E.H.
Computer based medical consultations, MYCIN
American Elsevier.
- (Sidner 83) : Sidner C.
Focusing in the compréhension of definite anaphora
In Computational Models of discourse, Ed. by M.Brady &
R.C.Berwick, The MIT Press, Cambridge, Massachussets.
- (Simmons 73) : Simmons R.F.
Semantic Networks : their computation and use for under-
standing english sentences.
In Computer models of thought and language,
Eds R.C.Schank & K.M.Colby, San Francisco:Freeman

- (Steel 84) : Steel S.
Simplifying recursive belief for language understanding.
ECAI-84, Pise.
- (Tesnières 53): Tesnière L.
Esquisse d'une syntaxe structurale.
Klinsksieck, Paris.
- (Vilnat 84) : Vilnat A.
How a system may be self-conscious.
ECAI-84, Pise.
- (Vilnat 85) : Vilnat A. et Sabah G.
La gestion du dialogue dans un système de questions-réponses.
Congrès AFCET-5ème Génération, Paris. (à paraître)
- (Webber 78) : Webber B.L.
A formal approach to discourse anaphora.
Technical Report 3761, Bolt, Beranek & Newman Inc.
- (Webber 83) : Webber B.L., Joshi A., Mays E., Mc Keown K
Extended natural language data base interactions
Computers & Mathematics with Applications, Vol.9, n°1.
- (Weiner 80) : Weiner J.L.
BLAH, A System Which Explains its Reasoning
Artificial Intelligence, Vol.15, p.19-48.
- (Weizenbaum 66) : Weizenbaum J.
ELIZA, A Computer Program for the Study of Natural Language
Communication Between Man and Machine, Comm ACM,
Vol. 9, n°1, p.36-45.
- (Wertz 78) : Wertz H.
Un système de compréhension, d'amélioration et de correction
de programmes incorrects.
Thèse de 3ème cycle, Université Paris 6.
- (Winograd 72) : Winograd T.
Understanding Natural Language
Academic Press, New-York.

